UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

TRABAJO FIN DE MÁSTER

TIC-TAC-TEA: Aplicación para recolección de datos biométricos y autorregulación emocional de personas con TEA

Máster en Ingeniería Informática

Autor: Cuadrado Manso, Javier
Tutor: Torrado Vidal, Juan Carlos

FECHA: Junio, 2017

Resumen

El Trastorno de Espectro Autista es un síndrome que afecta al desarrollo neurológico de las personas definido por deficiencias en las habilidades sociales y en la capacidad comunicativa, así como la aparición de comportamientos característicos. Una de las técnicas que existen para mejorar la comunicación de la persona o para que esta entienda mejor lo que se le quiere decir es el uso de pictogramas: Imágenes que acompañan de forma visual al lenguaje hablado o escrito. El proyecto TIC-TAC-TEA, impulsado por la Fundación UAM y la Fundación Orange, utiliza esa técnica para ayudar a las personas con autismo.

La aplicación TIC-TAC-TEA, instalada en un *smartwatch* Android, recopila datos biométricos utilizando los sensores de los que disponen esos dispositivos. Tras realizar una serie de cálculos con los datos obtenidos, en el caso de que se haya dilucidado que se ha entrado en un episodio de estrés, se mostrará una serie de imágenes y animaciones con el objetivo de que el individuo, al seguir sus instrucciones, sea capaz de finalizar la crisis.

A lo largo de este Trabajo de Fin de Máster se explicará en detalle el proceso de diseño y los detalles del desarrollo de esta aplicación para relojes Android. En este trabajo se detallarán los objetivos de la aplicación y sus requisitos además de explicar los problemas a los que se ha hecho frente para desarrollar la aplicación, como el manejo de los diferentes hilos, la recolección de datos biométricos o el protocolo implementado para comunicar datos entre reloj y el móvil.

PALABRAS CLAVE

Actividad, Android, aplicación, autismo, autorregulación emocional, datos biométricos, móvil, pictograma, protocolo de comunicación, sistema aumentativo y alternativo de comunicación, *smartwatch*, Wear.

Abstract

Autism Spectrum Disorder is a neurodevelopmental syndrome defined by deficits in social reciprocity, communication skills and by unusual restricted repetitive behaviours. One technique to improve autism people´s communication skills is using pictures to extends verbal language. TIC-TAC-TEA project, promoted by Fundación Orange and Fundación UAM, uses this technique to help people with autism.

Once is installed on a smartwatch, TIC-TAC-TEA application collects biometric data trough the sensors available on these devices. After doing some calculations using this data, the application will show a serial of static and dynamic images if a crisis is detected. The objective of showing this pictures, called regulation, is helping user to finish the crisis following the steps indicated on the images.

Throughout this memory It will be explained how this application for Android smartwatches was designed and implemented. In this memory the objectives and requisites of this application will be detailed, as well as other problems solved as how biometric data collection has been done or the design of the communication protocol between mobile and watch.

KEYWORDS

Activity, Alternative and Augmentative Communication, Android, application, autism, biometric data, communication protocol, emotional-self regulation, mobile, pictogram, smartwatch, Wear.

Glosario

Amilab: Laboratorio de inteligencia ambiental de la Universidad Autónoma de Madrid.

Android: Sistema operativo de Google disponible para distintos dispositivos como móviles, tabletas y relojes inteligentes.

Datos biométricos: Son aquellos datos de los procesos biológicos recopilados de los usuarios de la aplicación.

Estrategia: Secuencia de imágenes sobre la que se aplica una serie de reglas que determinan el modo transición entre estas o el tipo de animación que se puede utilizar para mostrar el tiempo trascurrido.

IPP: Instituto de Psicopediatría

Regulación: Conjunto de estrategias a mostrar en el reloj cuando se detecte una crisis en el usuario.

SDK: Kit de Desarrollo Software.

Smartwatch: Reloj inteligente que contiene un sistema operativo y puede estar equipado de sensores biométricos.

TEA: Trastorno de Espectro Autista.

TFM: Trabajo de Fin de Máster.

TIC-TAC-TEA: Es el nombre del proyecto y el nombre de la aplicación.

Wear: Versión del sistema operativo Android para relojes inteligentes.

Agradecimientos

En primer lugar, quisiera agradecer al tutor de mi proyecto, Juan Carlos Torrado, y a todo el personal del laboratorio Amilab el haberme permitido participar en este apasionante proyecto. Con la libertad que se me ha dado a la hora de realizarlo he podido aprender una cantidad impresionantes de cosas que espero que hayan servido en este proyecto.

Tampoco puedo dejar de mencionar a todo el personal del Instituto de Psicopediatría Dr. Quintero Lumbreras por habernos guiado y aconsejado a lo largo de todo el proyecto y por abrirnos las puertas de su institución.

También quiero agradecer a la Fundación Orange el haber impulsado este gran proyecto que seguro que será útil para muchísimas personas.

Además quiero agradecer a mi familia por haber estado ahí, apoyándome durante todo el Máster, sin ellos esto no habría sido posible.

Y por último, quiero dar las gracias a todo aquel que se haya interesado por este TFM, espero que su lectura le resulte atrayente y disfrute con ella.

Índice general de contenidos

Índice de figuras

1 - Introducción

1.1 - El trastorno de Espectro Autista

El trastorno de espectro autista hace referencia a una condición en el desarrollo neuronal definida por una serie de comportamientos característicos. Según el (DSM-5: Diagnostic and Statistical Manual of Mental Disorders), las principales características de este trastorno afectan a la comunicación e interacción social y la aparición de patrones de conductas restrictivas y/o repetitivas, Síndrome de Tourette (McGuire, 2014). Es llamado espectro tanto por la variabilidad de los problemas que aparecen como por el diverso grado que estos pueden tener.

Estas anomalías se presentan en edades tempranas del desarrollo de las personas, suelen ser evidentes a partir de los 3 años, y se mantienen a lo largo de toda su vida.

En el DSM-5, se han declarado los siguientes niveles de severidad del trastorno:

- Nivel 1: Requerimiento de soporte. En este grado de autismo existen dificultades en la interacción social y respuestas atípicas o infructuosas a propuestas de socialización de otras personas. En este nivel se considera que se tiene poca flexibilidad para salir de un entorno controlado y constante.

- Nivel 2: Requerimiento de un soporte sustancial. Incluye notables defectos en comunicación verbal y no-verbal, discapacidades sociales incluso con apoyo y limitaciones a la hora de iniciar interacciones sociales. Existen dificultades en entornos cambiantes y aparecen comportamientos restringidos o repetitivos de forma frecuente que interfieren en el desempeño de su rutina.

- Nivel 3: Requerimiento de soporte absoluto. En este grado se considera que existe déficits severos en comunicación verbal y no-verbal que provocan discapacidades en el funcionamiento del día a día, incapacidad de iniciar interacciones sociales y una mínima respuesta a interacción social por parte de terceros. Se es muy poco flexible con el entorno, con extrema dificultad de aceptar cambios.

Aunque actualmente no existe una metodología con la que eliminar este trastorno, se asume que la detección temprana de los distintos problemas es decisiva para la mejora de los niños, ya que se puede mejorar las habilidades funcionales y reducir las conductas disruptivas con una asistencia constante. Estas intervenciones intentan mejorar la vida de estas personas contrarrestando las deficiencias que provoca.

Desafortunadamente, debido a la variabilidad de las anomalías del trastorno no existe una metodología estándar que ayude a todos las personas por igual, aunque se está de acuerdo en que se debe centrar en las áreas de desarrollo alteradas, especialmente en las que refieren a competencias socio-comunicativas (Martos J, 2013).

Teniendo en cuenta que el canal preferente de procesamiento de la información en personas con autismo es el visual, uno de los métodos más comunes, y el utilizado en el centro en el que se probará la aplicación de este TFM, es el del uso de sistemas aumentativos y alternativos de comunicación, SAAC (Cudolá, 2016). Este sistema

combina la palabra con apoyos visuales, la cuál favorece la comunicación espontánea y funcional (Basil, 2010). Estos apoyos visuales son una serie de imágenes, llamados pictogramas, que relacionan conceptos con dibujos simples.

Uno de los problemas característicos del trastorno de espectro autista es el de las crisis (Kelli C. Dominick, 2006). Las crisis o episodios de estrés o ansiedad son situaciones en las que la persona tiene una saturación emocional o sensitiva y, al no poder hacer frente a estas sensaciones, tiene un estallido emocional que se puede mostrar de diversas maneras. En ocasiones estos episodios se reflejan como si fuese un enfado sin aparente sentido o berrinche, pero puede ser situaciones en las que esta persona se quede inmóvil o se infrinja lesiones a sí mismo.

Una de las formas de hacer frente a estas situaciones es intentar reconducir a la persona autista a una situación de normalidad. Como se ha explicado, no hay una metodología estándar, pero una forma frecuente de volver a la normalidad es utilizar los pictogramas, a veces como método de refuerzo o, como se intenta hacer con la aplicación TIC-TAC-TEA, como manera suficiente para autorregular sus emociones.

El objetivo final de esta aplicación es que las personas con autismo cuenten con una herramienta a su alcance, en sus relojes, que les ayude a autorregular sus emociones y superar fases de crisis.

1.2 - Motivación del proyecto

El proyecto TIC-TAC-TEA se presentó en el concurso "Soluciones tecnológicas aplicadas al autismo 2015" de la Fundación Orange y fue uno de los cuatro proyectos seleccionados para ser puestos en marcha (Orange, 2015). Con ayuda del Instituto de Psico-Pediatría Dr. Quintero Lumbreras (Lumbreras, 2017), se realizará la aplicación con el nombre del proyecto que ayude a las personas con autismo en el laboratorio de inteligencia ambiental (Amilab).

La aplicación TIC-TAC-TEA pretende ayudar a las personas autistas a autorregular sus emociones mostrando una serie de imágenes en un reloj Android. Conectado por Bluetooth a un *smartphone*, este reloj contará con una serie de sensores que recopilarán datos biométricos del usuario del dispositivo. Haciendo uso de esos datos, se realizarán una serie de cálculos sobre los datos obtenidos en los últimos segundos para dilucidar si se ha entrado en un estallido emocional. En el momento en el que se detecte una situación de crisis, se mostrará una serie de imágenes. Esas imágenes o pictogramas, serán elegidas a través de la herramienta aplicación móvil, desarrollada en otra aplicación del proyecto TIC-TAC-TEA. Tras enviarse al reloj, serán mostradas en la aplicación para *smartwatch* Android cuando se detecte una crisis. Esta aplicación para *smartwatch* es el objetivo a desarrollar en este TFM, así como el protocolo de comunicación de datos entre móvil y reloj.

1.3 - La regulación emocional

Como se ha comentado, el objetivo de esta aplicación es que las personas con autismo cuenten con una herramienta a su alcance en sus relojes que les ayude a regular sus emociones y superar fases de crisis. Esa herramienta se basa en mostrar una **regulación** cuando el reloj detecte una situación de estrés, la cuál se repetirá una y otra vez hasta que se detecte que se ha terminado la crisis.

La **regulación** es una secuencia de **estrategias**, que son a su vez una serie de imágenes a las que se les llamará **pasos**, y sobre los que se aplicará una serie de reglas establecidas en la **estrategia** a la que pertenecen. La regulación muestra su secuencia de estrategias en el orden configurado hasta que esta llegue a su fin.

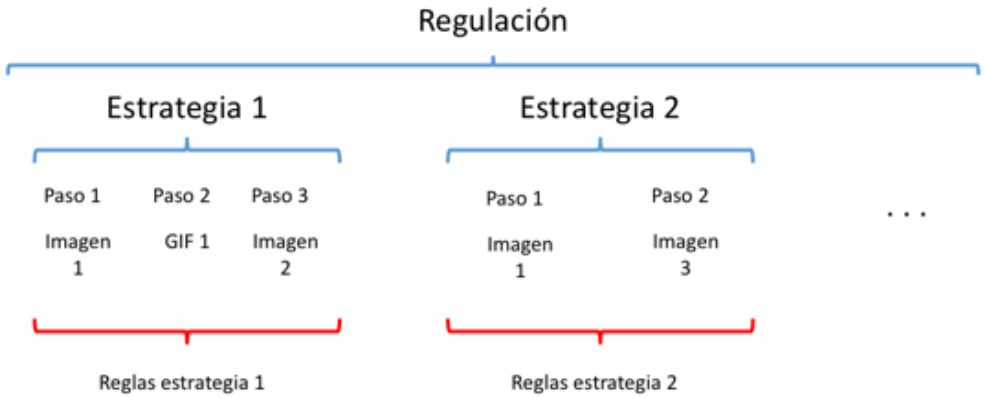

Ilustración 1: Composición de una regulación

Las **estrategias** muestran los **pasos**, que pueden ser imágenes estáticas o dinámicas, GIFs, en el orden que esté configurado en estas y sobre los que se aplica las reglas seleccionadas en cada estrategia. Las reglas determinan si la transición de un paso al siguiente se puede realizar de forma manual, es decir, por un toque en la pantalla, o si por el contrario se realizará de forma automática pasado un tiempo.

Las reglas también determinan cuanto tiempo durará la **estrategia** en total o cada uno de sus **pasos** en pantalla si la transición no es manual. Así mismo, estas indicarán qué tipo de animación existe para indicar el paso del tiempo si la transición es automática: Una animación de una barra de progreso que rodee la pantalla o un efecto que *inunde* la pantalla hasta que llegue el fin de la **estrategia**.

Por último, los **pasos**, como se ha dicho antes, son cada uno una imagen estática o dinámica, llamados también **pictogramas**. Estos **pictogramas** se obtienen de la web de Portal Aragonés de la Comunicación Aumentativa y Alternativa, ARASAAC, (Aragón, 2017), de la cuál se han extraído las imágenes que utilizan en el centro diariamente. También se podrán tomar mediante una cámara integrada en algún móvil.

Además de pasos de **pictogramas**, existen algunos pasos especiales, que mejoran la interacción del usuario con la aplicación.

Uno de ellos es un selector, el cuál muestra una lista con una serie de estrategias entre las que elegir una: la siguiente que se va a realizar. Sea cual sea la estrategia escogida, el resto de la regulación seguirá mostrándose en el orden establecido.

Otro de los pasos especiales es el que transforma una estrategia en un carrusel, es decir, una secuencia de imágenes alineadas de izquierda a derecha. Este paso recogerá todas las imágenes estáticas de la estrategia y las mostrará una tras otra en un carrusel interactivo que permitirá al usuario pasar las imágenes hacia delante y hacia atrás, hasta que se agote el tiempo estipulado para la estrategia.

Y por último, el paso de confirmación, en el que se pedirá al usuario que confirme si está tranquilo pulsando en un tic verde o rojo. En caso de que seleccione la opción afirmativa, el tic verde, y se detecte que sus medidas biométricas son normales, se dará por terminada la regulación, pudiendo mostrar una última imagen a modo de recompensa si así lo dictamina la estrategia. En caso contrario, se continuará con la siguiente estrategia o se finalizará la regulación si ha llegado a su fin.

La **regulación** se fabricará en la aplicación móvil de TIC-TAC-TEA y se transmitirá al reloj junto con todos los pictogramas y la configuración necesaria para montar la regulación, sus estrategias y el orden que siguen y qué pictogramas usa cada una. Las estrategias y las imágenes de cada una pueden repetirse, sin que esto repercuta en duplicidades de los recursos.

1.4 - Datos biométricos

Para que la herramienta funcione correctamente, es necesario que el reloj en el que se va a desplegar la aplicación cuente con los sensores que se han considerado básicos para recopilar datos biométricos (Hollocks, 2014).

Estos datos biométricos se utilizarán con dos fines. El primero será la creación de un perfil que dictamine si el usuario de la aplicación está en una situación de crisis o estrés. Este perfil estará integrado en la versión final de la aplicación para reloj. El segundo es el de recolectar los datos de los últimos segundos y, tras realizar unos cálculos, compararlos con el perfil añadido a la aplicación. A estos últimos segundos en los que se recogerán datos sobre los que se van a realizar los cálculos se les denomina ventana de tiempo. Mientras la aplicación que se va a desarrollar esté activa, se recopilarán de forma continua los datos biométricos y al finalizar cada ventana de tiempo se compararán con el perfil creado.

Como punto de partida, se ha establecido que las medidas a recopilar son el pulso cardíaco (por lo que es necesario un pulsómetro) y el movimiento del sujeto. Este último se descompondrá en tres medidas: Aceleración (para lo que hará falta un acelerómetro), giro (se necesitará un giroscopio) y pasos detectados. Todos estos sensores deberán estar integrados en el reloj que se utilizará en las pruebas y, a ser posible, en los relojes que utilicen los usuarios finales.

1.5 - Objetivos del proyecto

Este proyecto pretende cumplir con las metas comentadas en apartados anteriores, que es crear una aplicación para *smartwatch* que recopile datos biométricos de personas autistas y otra que utilice esos datos para detectar crisis y mostrar una serie de pictogramas por pantalla. Para alcanzar este fin, se han de establecer unos objetivos previos que han de ser cumplidos:

- Analizar los *smartwatches* y las herramientas existentes: El primer paso será analizar qué relojes son aptos para el desarrollo de este proyecto y ver qué tipo de aplicaciones existen para personas que tienen este trastorno.

- Diseñar una herramienta de recopilación de datos biométricos: Esta aplicación para relojes Android será la encargada de recopilar los datos biométricos necesarios para crear un perfil con el que determinar si la persona ha entrado en un episodio de estrés o ansiedad. Este perfil, una vez se haya definido, se añadirá a la aplicación.

- Diseñar una herramienta de autorregulación: El perfil que se cree con los datos obtenidos se añadirá en una aplicación cuya función principal es mostrar una serie de imágenes, pictogramas, para que el usuario sea capaz de reconducir sus emociones.

- Diseñar un protocolo de transmisión de datos: Los pictogramas que se muestran en la aplicación para reloj serán personalizados para cada persona, por lo que deberán ser escogidas desde la herramienta de autor del móvil. Por tanto, para enviar estas imágenes o recibir los datos recopilados en el reloj, se diseñará un protocolo que envíe y reciba datos y archivos entre ambos dispositivos.

1.6 - Estructura de la memoria

Este Trabajo de Fin de Máster está estructurado de la siguiente forma:

- Capítulo 1: Sirve como introducción al trabajo desarrollando las motivaciones para realizar el proyecto y los objetivos a cumplir. También se ha realizado una pequeña introducción a qué es el autismo, qué es una **regulación** y los datos biométricos que se van a recopilar.

- Capítulo 2: Aquí se describe el sistema de Android y su entorno, así como los detalles de la conexión entre los *smartwatches* y móviles Android. También se explican algunos puntos que se han de conocer al leer este trabajo como qué es una actividad o qué tipos de hilos existen en Android. Además se ha realizado un estudio de mercado de otras aplicaciones para personas con autismo y los relojes disponibles en el mercado en el momento de empezar el proyecto.

- Capítulo 3: En él se detalla en qué consiste el proyecto TIC-TAC-TEA.

- Capítulo 4: En este capítulo se describe la primera aplicación desarrollada: La aplicación de recopilación de datos biométricos.

- Capítulo 5: Se detalla cómo se ha desarrollado la segunda aplicación de TIC-TAC-TEA, que toma como base lo realizado en la primera aplicación.

- Capítulo 6: Es el final del trabajo, donde se recogen las conclusiones extraídas a a lo largo de este proyecto y se plantean algunos posibles objetivos como trabajo futuro.

2 - La plataforma Android

Android es un sistema operativo que está basado en Linux y sobre el que corre un *framework* Java con el que se pueden desarrollar aplicaciones para este sistema. Android se encuentra disponible en un amplia variedad de dispositivos, entre los que se incluyen móviles (*smartphones*) y relojes inteligentes (*smartwatches*), en este último es llamado Android Wear, por ser un dispositivo que se lleva encima.

Android no es el único sistema operativo que funciona tanto en móviles y como en relojes, iOS y Tizen también cuentan con versiones para los dos dispositivos. Las dos razones de haber escogido la plataforma de Android frente a otras alternativas son dos: diversidad y popularidad.

Por un lado, Android cuenta con una gama mucho más amplia de dispositivos que iOS. Mientras que el sistema operativo de Apple solo funciona en sus propios terminales, Android cuenta con miles de modelos de móviles y decenas de relojes, teniendo estos una gran variedad de precios, desde más asequibles hasta los más caros. En el caso de Tizen, sus terminales apenas están disponibles fuera de Corea del Sur.

Por otro lado, en muchos países de todo el mundo Android cuenta con una gran cuota de uso. En España, según los análisis de (Kantar, 2016), la tasa de adopción de móviles con Android es del 91% frente al 8% de Apple, lo que significa que los usuarios prefieren dispositivos Android por encima de los de Apple.

Sabiendo esto, el sistema operativo elegido es Android. ¿Y con qué versión?. Uno de los problemas que tiene Android es de la fragmentación causada por la falta de actualizaciones del sistema debido a que estas dependen de los fabricante. Por tanto, hay que escoger aquella versión que dé un máximo servicio a los clientes y a su vez sea lo más moderna posible. Siguiendo las datos que proporciona Android (Google, 2016), en el momento del inicio del desarrollo, la mejor versión para desarrollar era la 4.4 porque contaba con el 99% de compatibilidad. Al subir versiones a la versión 5.0 se perdía casi un 35%, por lo que se decidió utilizar la 4.4, la cual era a su vez la versión mínima que soportaba conexiones con Android Wear. Por su parte, aunque las actualizaciones de Android Wear corren a cargo de Google, se ha podido comprobar que no todos los relojes se actualizan, por tanto la versión mínima escogida para los relojes es la primera, Android Wear 1.0.

2.1 - Estados de conexión móvil-reloj

A la hora de manejar aplicaciones Android que tengan una conexión con su homóloga en un reloj de Android Wear, es conveniente conocer cómo se conectan.

En primer lugar, es importante saber que la conexión del móvil con el reloj no se puede realizar desde cualquier aplicación. Solo la aplicación de Android Wear puede conectar o desconectar un móvil y un reloj. Esto implica que la aplicación a desarrollar debe estar preparada para identificar el reloj con el que se esta comunicando, por si hubiese varios disponibles, o siquiera si hay algún reloj. Además la conexión entre ambos dispositivos únicamente se puede realizar por Bluetooth o por Wifi si ambos están en la misma red y si el reloj admite este tipo de conexión. La conexión Wifi del reloj únicamente sirve para conectarse con el móvil, puesto que las APIs de conexión a la red están limitadas.

Por otra parte, hay que conocer los tres estados de conexión entre un móvil y un reloj.

- Emparejado. Un móvil puede ser emparejado con varios relojes, pero no al revés. El proceso de emparejamiento solo incluye la identificación inicial del reloj, la instalación de las aplicaciones y la configuración del sistema. Hecho el emparejamiento, el reloj no tiene porqué seguir estando en el alcance del móvil. Para que un reloj se empareje con otro móvil, debe ser restaurado de fábrica, puesto que debe borrar todas las aplicaciones y la información que contiene del móvil.

- Conectado. Como se ha dicho, un reloj puede estar fuera del alcance del Bluetooth o Wifi del móvil. Solo si entra en el alcance del Bluetooth o en la misma red que el móvil se realizará la conexión entre ambos dispositivos. Estar conectados únicamente implica que la aplicación Android Wear actualizará los datos del reloj y quizás el sistema operativo, cualquier conexión entre aplicaciones no está habilitada.

- Sincronizado. Mediante la aplicación de Android Wear se puede elegir cuál de los relojes que están conectados al móvil se va a sincronizar. Solo puede existir un reloj sincronizado al mismo tiempo. Mientras el reloj esté sincronizado, y por tanto conectado dentro del área del móvil, se permitirán las comunicaciones entre las aplicaciones. Las aplicaciones solo pueden comunicarse entre sus homólogas, lo que significa que TIC-TAC-TEA Mobile puede comunicarse exclusivamente con la versión Wear y viceversa, pero estas no podrán transmitir datos a otras aplicaciones.

2.2 - Actividades

Otro punto a tener en cuenta es cómo funcionan las aplicaciones Android y es mediante actividades. Dicho de forma simple, una **actividad** es una pantalla y una clase que gestiona el ciclo de vida de la actividad además de recibir los eventos producidos por la interacción del usuario con la pantalla.

Cada actividad tiene por defecto una pantalla, descrita generalmente en un fichero XML, que se muestra al usuario y una sola clase que gestiona toda su lógica. Lo normal es que el usuario solo vea una actividad al mismo tiempo, aunque existen excepciones, como los diálogos o los menús desplegables, que se pueden mostrar encima de esta.

Hay que destacar las palabras "por defecto" del párrafo anterior, puesto que en los relojes cada pantalla puede estar descrita por un XML genérico y un XML específico de la forma del reloj. El XML genérico hace referencia al XML específico según la forma del dispositivo, teniendo en total dos específicos: Uno para relojes con pantalla circular y otro para pantallas cuadradas.

El ciclo de vida de las actividades viene determinado por la forma de iniciar o terminar cada una de ellas, es importante conocer este ciclo de vida ya que condiciona a las funciones que se van a llamar. Si una actividad se está iniciando desde 0, se llamará a onCreate, donde se crearán los componentes, y onStart, donde se pueden hacer modificaciones sobre los componentes en función de la lógica, después de onCreate u onRestart. onResume es una función llamada justo al hacerse visible la actividad. onPause y onResume son funciones utilizadas cuando la actividad es parcialmente tapada por otra o cuando esta se "congela" para mostrar otra (como un diálogo modal). onStop es llamada cuando la actividad se va a ocultar y de ahí se llamará a onRestart si esta vuelve a hacerse visible o a onDestroy si la aplicación se cierra o la actividad no es necesaria en memoria.

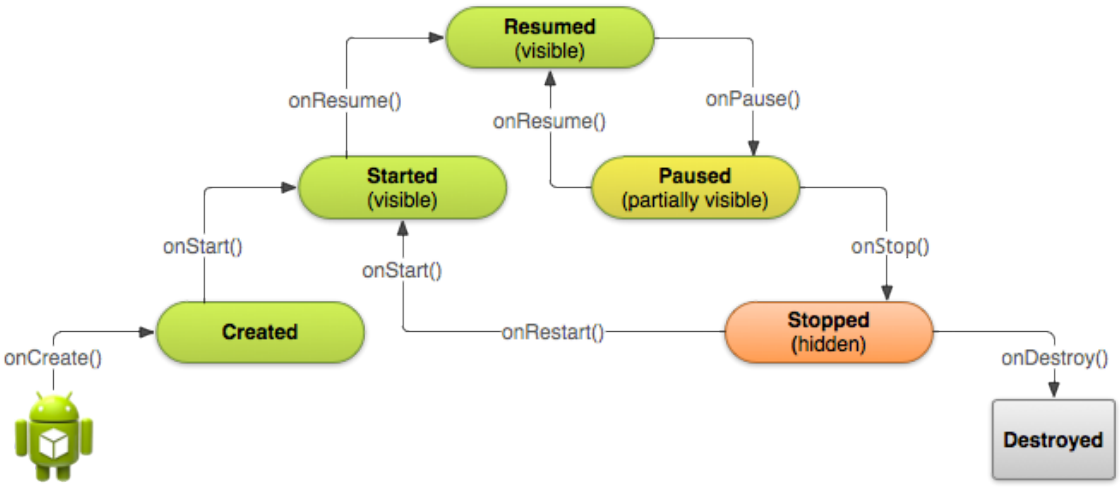

Ilustración 2: Diagrama del ciclo de vida de una actividad

El uso de las funciones de las actividades ha sido clave para algunas de las funcionalidades de la aplicación para reloj, como iniciar las mediciones de datos biométricos solo una vez, sin que se dupliquen las medidas.

2.3 - Tipos de hilos y multitarea

Las aplicaciones Android tienen principalmente dos tipos de hilos: El principal y los secundarios.

El hilo principal es el hilo que maneja el ciclo de vida de las actividades que conforman la aplicación, el encargado de pintar la interfaz gráfica de la actividad visible, el que la actualiza según la lógica interna de la aplicación y el que recoge las interacciones del usuario con la interfaz, como puede ser tocar un botón. Este hilo tiene un comportamiento especial, puesto que en ningún caso puede ser parado o pausado con las funciones de espera *wait*, *sleep* o similares. Además, el propio sistema operativo supervisa que el hilo no sea capturado por ninguna función durante un intervalo de más de 5 segundos, puesto que es necesario que el hilo repinte la interfaz o reciba interacciones del usuario de forma continua. Por este motivo, si el sistema operativo detecta una captura más larga de lo permitido, considerará que la aplicación ha dejado de funcionar y la cerrará, junto a cualquier hilo secundario que hubiese.

Sabiendo esto, existen operaciones que no se pueden realizar en el hilo principal por tener una carga de tiempo superior a este intervalo, ser resueltas en un tiempo desconocido (como puede ser el tiempo entre una petición a un servidor y su respuesta) o simplemente por querer utilizar expresamente las capacidades de los procesadores de varios núcleos. Para cualquiera de estos casos existen los hilos secundarios, que se pueden clasificar en: Tarea asíncrona, escuchas (*listener*) de evento, servicios propios, servicios de Android e hilos (*thread*) de Java.

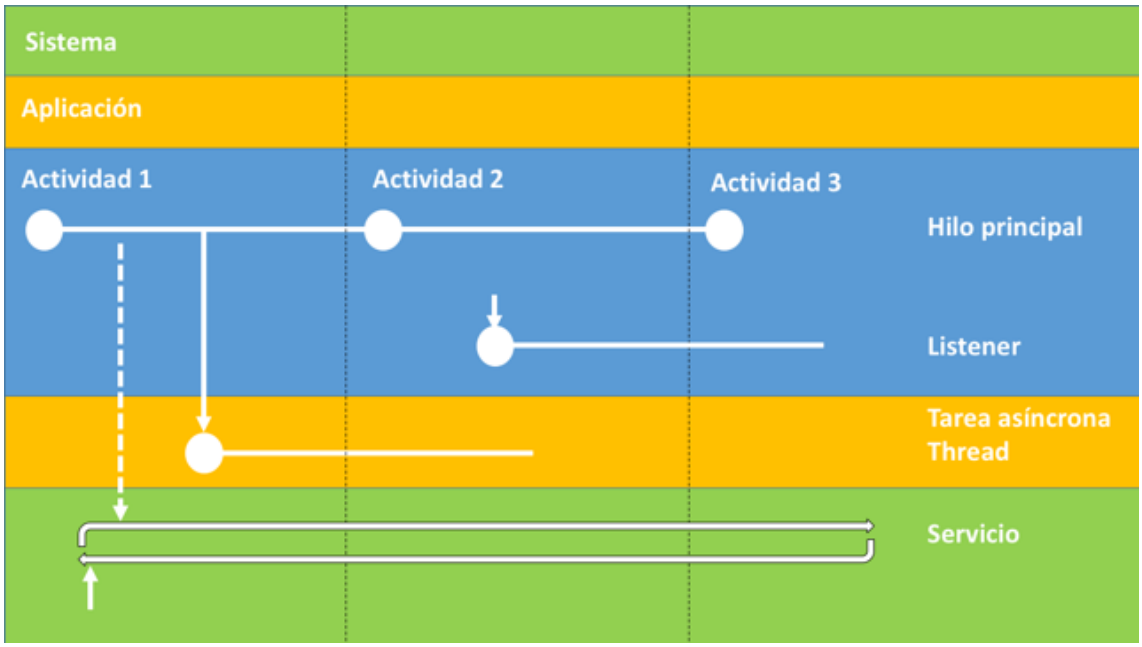

Ilustración 3: Diagrama de hilos

Las tareas asíncronas son hilos en segundo plano pensados para ejecutar código de forma asíncrona al hilo que la ha creado. Su objetivo es realizar pequeñas tareas con un final definido y son independientes al ciclo de vida de la actividad o proceso que lo ha lanzado. Precisamente por ser independiente al proceso que lo ha lanzado, estas tareas deben tener un final definido y no ejecutarse eternamente, puesto que solo se paran si el sistema detiene todos los hilos de la aplicación como cuando esta es desinstalada. El código de

las tareas asíncronas viene definido en tres funciones de la clase que extiende *AsyncTask*, una previa al código a ejecutar, otra que contiene el código principal de la tarea y otra ejecutada cuando se termina la tarea. Aunque en los móviles las tareas asíncronas pueden lanzarse libremente, en las primeras versiones de Android Wear y los primeros relojes existe la gran limitación de únicamente poder lanzar una tarea asíncrona simultáneamente por aplicación. En caso de intentar lanzar otra tarea asíncrona, este hilo morirá antes de ejecutar la primera instrucción. Por este motivo se han buscado otras alternativas.

Los servicios son un tipo de hilo secundario o subproceso (según se utilice) diferente. Aunque el número de servicios de una aplicación es indefinido y su ciclo de vida es independiente al del resto de la aplicación, se diferencia de las tareas asíncronas en que no se puede lanzar servicios duplicados. La forma de identificar los servicios es mediante el nombre de la clase que extiende la clase *service*. Mientras que clase que hereda de *Asynctask* se puede lanzar tantas veces como se quiera, las clases de los servicios solo se pueden lanzar una vez al mismo tiempo. Esto es especialmente útil para el objetivo de un servicio: ser ejecutado de forma casi indefinida, parando únicamente cuando se necesite.

Otro tipo de hilo es el generado con los *listener* (escuchas). En ocasiones, es conveniente hacer caso a determinados eventos del sistema operativo (como el aviso de que hay poca batería) o estar atento ante eventos lanzados por la propia aplicación. La API de Android permite lanzar eventos de aplicación y escucharlos (tanto a los de aplicación como a los del sistema operativo). Al recibir el evento que ha sido lanzado, se genera un hilo que ejecuta el código asignado a la escucha del evento. La asignación de código con eventos se puede realizar registrando la escucha programáticamente o utilizando el fichero de configuración de la aplicación Android (al que se conoce como manifiesto). No hay que confundir estos *listener* con aquellos que manejan los eventos de los elementos de la interfaz gráfica, como puede ser pulsar un botón, ya que ese evento es manejado por el hilo principal y su existencia está ligada al ciclo de vida de la actividad.

Una mezcla de los servicios y los *listeners* es el de los servicios de Google. Estos servicios son lanzados cuando se recibe un evento especial de Google, como puede ser un mensaje enviado desde un reloj, y se ejecutan hasta que se considere oportuno. Al igual que los servicios, los de Google son independientes al ciclo de vida de la aplicación, aunque estos no pueden ser ejecutados por el resto del código. Es más, la aplicación ni siquiera es consciente de cuándo se están ejecutando los servicios de Google.

Para finalizar, fuera del SDK de Android, está el hilo (*thread*) de Java. Aunque no debría ser utilizado ya que se dispone de los anteriores tipos de hilos incluidos en el SDK, esta clase ha sido útil para esquivar la limitación de la tarea asíncrona de los relojes.

Hilo	Iniciado por:	Finalizado por:	Tipo de tareas:
Principal	Aplicación	Aplicación	Pintar interfaz Interacción usuario
Tarea asíncrona	Código	Código Aplicación	Tareas cortas
Listener	Evento	Código Actividad Aplicación	Código asociado a evento
Servicio	Código Sistema	Código Sistema	Servicios push. Notificaciones
Thread	Código	Código Aplicación	Tareas cortas

Ilustración 4: Comparación de hilos

2.4 - Estudio de mercado

En este apartado se va a realizar un análisis del estado del mercado para las aplicaciones existentes en este área y los dispositivos disponibles para los usuarios.

2.4.1 - Aplicaciones

El mercado de aplicaciones para personas con autismo, y en especial para niños, está lleno de alternativas y no es difícil encontrar una que se adapte a las necesidades de cada uno, por lo que solo se van a comentar algunas de las soluciones encontradas.

Como se ha comentado antes, las aplicaciones pretenden apoyar o mejorar aquellas áreas donde el autismo hace mella, por lo que la búsqueda se ha dividido en categorías correspondientes al área.

Comunicación

Son aplicaciones que apoyan a la persona autista a la hora de comunicarse con otras personas o viceversa o que pretenden mejorar sus habilidades comunicativas. En esta categoría se incluyen aquellas que utilizan la metodología de comunicación SAAC.

- Proloquo2Go (Proloquo2Go): Se trata de una aplicación para iPhone e iPad que permite a la persona con autismo construir frases, que serán leídas en voz alta por la aplicación, utilizando pictogramas. La versión de iOS soporta el Apple Watch.
- Voice4u (Voice4uaac): Es una aplicación que, además de construir frases a partir de pictogramas, lee el texto que se le introduzca y lo traduce al idioma que se quiera.
- Words in Pictures (fingertalksapp): App pensada para padres, profesores y terapeutas que necesitan hablar con niños o adultos con autismo utilizando pictogramas.
- LetMeTalk (letmetalk): Similar a la anterior, permite la comunicación entre padres/profesores y personas con autismo y viceversa utilizando pictogramas. Destaca entre las otras por utilizar los pictogramas de ARASAAC, la página de la que se sacarán los pictogramas para la aplicación TIC-TAC-TEA y que son utilizados diariamente en el Instituto de Psico-Pediatría Dr. Quintero Lumbreras.

Aprendizaje

Son aplicaciones que ayudan a las personas con autismo a aprender. La mayoría de estas aplicaciones apoyan la enseñanza del lenguaje.

- Palabras Especiales (specialiapps): Es una aplicación utilizada para enseñar a los niños a reconocer palabras escritas, utilizando imágenes y sonidos.
- Phonics Keyboard (therapy-box): Es un teclado diseñado para los que empiezan a deletrear y a leer, con imágenes y sonidos asociados a cada letra.
- Accesible Literacy Learning (tobiidynavox): Sirve para ayudar a los estudiantes sin capacidad verbal a aprender a leer. Divide la lectura en habilidades:

correspondencia letra-sonido, combinar sonidos, segmentación de fonemas, decodificar palabras simples, reconocer palabras y lectura compartida.

Herramientas de apoyo

Son aplicaciones que apoyan de alguna forma a la persona autista.

- PictogramAgenda (lorenzomoreno): Sirve para planificar las actividades a realizar utilizando pictogramas. Esta aplicación sea ha destacado porque la planificación de las actividades del día en una agenda mediante pictogramas es parte de la rutina seguida en el Instituto de Psico-Pediatría Dr. Quintero Lumbreras.
- Magnus Cards (magnusmode): Es una aplicación que muestra secuencias de pasos a seguir relacionados con hábitos del día a día utilizando tarjetas con imágenes y una descripción de cada paso.
- This for That Visual Schedules (pixelationlabs): Se utiliza para realizar planificaciones visuales mediante fotos y una pequeña descripción. Indica el progreso de las tareas y muestra una imagen con el resultado que se debe obtener.

Emociones y comportamiento social

Se trata de aplicaciones que pretenden ayudar a reconocer y regular emociones en función de la situación y a relacionarse con el entorno.

- How are you? (tapp-mobile): Es una aplicación que enseña las características de cada emoción para reconocerlas y reproducirlas.
- CRUZ ROJA – Prevención de accidentes y primeros auxilios (dadacompany): Enseña a los más pequeños a mejorar su seguridad a través de la prevención de accidentes y aprendizaje de nociones básicas de primeros auxilios utilizando imágenes a modo de cuento.
- Vamos a aprender emociones (myeverydayspeech): Sirve para aprender emociones en forma de tarjetas y con juegos interactivos: adivinar la emoción, dos juegos de emparejar emociones y un juego de memoria.
- GoSequencing (smartyearsapps): Útil para trabajar las habilidades de secuenciación de juegos, comidas, higiene personal, naturaleza, tareas domésticas, etc.

Recursos para adultos

Son herramientas que permiten a padres o tutores ayudar o implicarse con las personas autistas.

- Historias especiales (specialiapps): Sirve para crear historias a partir de las imágenes y sonidos de los que dispone la aplicación.
- Pictello (assitiveware): Es una aplicación para crear historias y libros visuales con voz. Cada página de los libros puede contener una foto, un vídeo o un texto que se puede cargar a la aplicación.

- Autism & Beyond (autismandbeyond.researchkit.duke): Es una aplicación para recopilar datos. Analiza las expresiones faciales de los niños cuando estos visualizan videos para reconocer sus reacciones.

Ocio

Se trata de juegos que entretienen a la vez que pretenden ayudar a las personas autistas.

- El sueño – Hablando con el arte (fundacion orange): Una app que busca acercar el mundo del arte y la cultura, a través de la educación emocional y la creatividad, a las personas con autismo.
- Busy Shapes (edokiacademy): Sirve para trabajar el manejo objetos sencillos mediante una serie de ejercicios en los cuales deben colocar figuras sencillas en el lugar adecuado. La app responde a las acciones del niño y lo va guiando y motivando positivamente. Este tipo de juegos es destacado porque es una actividad frecuente en niños con autismo, realizada también en el Instituto de Psico-Pediatría.
- Tic Tac Time (edokiacademy): Una serie de minijuegos para que los niños se familiaricen con las manillas del reloj y aprendan a leer la hora.

Viendo las anteriores categorías, la aplicación TIC-TAC-TEA puede estar en la de emociones, puesto que pretende que la persona con autismo sea capaz de reconducir sus emociones al entrar en una situación de estrés.

Resulta sorprendente comprobar que la tecnología de los *smartwatches* apenas ha sido utilizada, estando las pocas aplicaciones que existen casi en exclusiva para la plataforma de Apple Watch. Esto es debido a que Apple Watch es el *smartwatch* más popular en ventas y más reconocido de todos, lo que significa que es la opción preferida por muchos desarrolladores pese a sus elevados precios. Por otra parte, resulta especialmente difícil encontrar esas aplicaciones que sí utilizan los relojes inteligentes como recurso, siendo los de la Apple Store los únicos visibles.

2.4.2 - Relojes

En el momento en el que el proyecto se inició, la variedad de relojes inteligentes era muy limitada. Partiendo de que se va a necesitar recopilar algunos datos biométricos, siendo el pulso el más básico, quedan descartados todos aquellos relojes que no tienen un pulsómetro. Por tanto, los relojes accesibles en mercado español, puesto que algunos no han llegado o no son fácilmente accesibles, que cumplen la característica del pulsómetro son los siguientes:

- Motorola 360
- LG G Watch R
- LG Watch Urbane
- Samsung Gear Live

En un principio se disponía en el laboratorio del Amilab de dos relojes: Un Motorola 360 y un LG G Watch R. Al principio se utilizarían estos relojes para el desarrollo, pero

llegado el momento de empezar a recopilar datos en el Instituto de Psico-Pediatría, fue necesario comprar varios más, por lo que había que sopesar las ventajas y desventajas de los que estaban disponibles.

Al inicio se escogió el Motorola 360 para iniciar el desarrollo de la aplicación, puesto que es el reloj impulsado por Google, sin embargo, su anticuado Hardware y su poca potencia limitaba las posibilidades en exceso, quedando descartado tras unas semanas de uso.

El LG G Watch R suponía una clara mejora respecto al modelo de Motorola por tener un Hardware más moderno, pero al ser un modelo residual que ya no se vende en tiendas por tener un remplazo se acabó descartando su compra.

El modelo de Samsung era un caso similar al del anterior LG, era un modelo antiguo pero que se seguía vendiendo.

Finalmente, el modelo elegido fue el que reemplazó al primer LG, el LG Watch Urbane, el cuál se vende en tiendas y es un reloj totalmente actual. Tras adquirirlo, se ha utilizado este modelo para probar la aplicación tanto en el laboratorio como en el instituto.

3 - El proyecto TIC-TAC-TEA

El proyecto TIC-TAC-TEA, uno de los ganadores del concurso "Soluciones tecnológicas aplicadas al autismo 2015" de la Fundación Orange se basa en una aplicación para *smartwatches*. Esta aplicación pretende utilizar los nuevos relojes inteligentes para conseguir que las personas con autismo que tienen episodios de crisis sean capaces de superarlos por sí mismas utilizando para ello los pictogramas que ellos mismos reconocen en la pantalla del reloj. Usando los datos biométricos que la aplicación recopile y los datos obtenidos de observar a los sujetos de estudio que utilizan la aplicación de autorregulación, se realizará una investigación que analizará la eficacia de este método.

La aplicación TIC-TAC-TEA está compuesta de tres partes: La herramienta de autor para dispositivos móviles, la aplicación de autorregulación emocional para *smartwatch*, objetivo de este TFM, y el perfil de detección de crisis incluido en dicha aplicación, que será incluido en la aplicación una vez terminado.

Estos tres componentes forman parte de la misma aplicación Android, TIC-TAC-TEA, la cual se distribuirá mediante Google Play. La misma aplicación empaquetará los tres componentes. Al instalar la aplicación en el dispositivo móvil, automáticamente se instalará la aplicación en el reloj si este está conectado.

Como se ha comentado anteriormente, el objetivo de este TFM es el desarrollo de la aplicación para Android Wear (el reloj) y el protocolo de transmisión de datos entre dicho reloj y el móvil al que estará conectado. Para ello, se ha desarrollado también una aplicación de pruebas para móvil con el fin de comprobar el correcto funcionamiento del protocolo, si bien la aplicación móvil será sustituida por la herramienta de autor cuando esté disponible. En cuanto al perfil que analiza los datos recopilados y dicta si existe crisis o no formará parte de la aplicación Wear a través de una librería incorporada en la app una vez esté finalizado.

Por otra parte, dado que se necesita realizar un perfil con el que poder detectar crisis en función de los datos biométricos recopilados, se ha decidido dividir la aplicación en dos fases. La primera tiene como prioridad recopilar datos biométricos como las pulsaciones o el movimiento realizado por el usuario, guardarlos y retransmitirlos. Una vez desarrollada esta versión, el objetivo es usar esta aplicación para recopilar datos mientras se observan y anotan manualmente los momentos en los que se detecte crisis para poder realizar una correlación entre ambos sucesos y así obtener un umbral que determine si hay crisis o no. La segunda fase, la aplicación de regulación emocional, será la que se juntará con la versión móvil para subirse a Google Play, es la que tendrá el resto de funcionalidades. El motivo de realizar esta división no es otro que dedicar tiempo a recolectar datos con la primera aplicación mientras se desarrollan otras funcionalidades en la segunda.

Además hay que tener en cuenta que las funcionalidades concretas de la aplicación de la segunda fase no han estado definidas desde el principio, muchas de estas se han incluido a medida que se iban probando las anteriormente implementadas o a medida que se ajustaban a lo que los relojes pueden ofrecer. Parte de este trabajo es averiguar cómo ajustar las necesidades descritas por el personal de Instituto de Psico-Pediatría Dr. Quintero Lumbreras en una aplicación para un reloj.

4 - Fase 1: Aplicación de Recopilación de datos

4.1 - Funcionalidad requerida

Como se ha comentado en el apartado anterior, esta aplicación únicamente recolecta datos como pulsaciones, movimientos y giros de muñeca y pasos. Estos datos se han de guardar junto con la fecha y hora de recopilación de cada datos. Además, para poder procesar los datos y dada la imposibilidad de acceder manualmente a los ficheros, la aplicación debe contar con la característica de transmitir los ficheros a un lugar desde el que poder leer los datos manualmente, como es el móvil emparejado con el reloj.

4.2 - Requisitos

En este apartado se va a especificar los requisitos de la aplicación de recopilación de datos. Hay que tener en cuenta que, tanto en esta aplicación como en la de regulación emocional, aunque se pedían unas ciertas funcionalidades, la formalización de requisitos se ha hecho sobre la marcha y a medida que se iban detectando las posibilidades y limitaciones de la plataforma de Android Wear. Por otra parte, esta flexibilidad también se debe a que muchas de las funcionalidades se han especificado en función de las distintas pruebas que se han ido realizando durante el desarrollo de las apps. En este documento se detallarán los requisitos que tendrá finalmente la aplicación.

4.2.1 - Requisitos funcionales

Para empezar, el sistema contará con un módulo dedicado a la recolección de datos a partir de los sensores que dispone el reloj:

- RS1: El reloj deberá recopilar los datos proporcionados por el sensor cardiaco. Esos datos serán la media de pulsaciones del periodo de tiempo tras la última medición, tiempo establecido por el fabricante del sensor, así como la precisión de la medida tomada. Esta precisión irá desde 0 (No se están tomando las pulsaciones bien porque no hay contacto con el cuerpo del usuario) hasta 3 (las medidas parecen estar realizándose con total precisión).
- RS2: El reloj deberá recopilar los datos proporcionados por el detector de pasos. La medida se tomará cada vez que se realice un movimiento interpretado como un paso. Esta medida irá acompañada de su precisión, siendo 0 (No se están tomando medidas), 1 (El movimiento apenas se asemeja a un paso), 2 (El movimiento se asemeja a un paso pero no es seguro) o 3 (El movimiento probablemente sea el propio de un paso). Nótese el *probablemente*, puesto que se detecta un paso cuando hay un balanceo del brazo de adelante hacia atrás y viceversa.
- RS3: El reloj deberá recopilar los datos proporcionados por el sensor de movimiento, acelerómetro, el cuál devolverá la cantidad de movimiento detectada en coordenadas X, Y, y Z cada cierto tiempo. La medida deberá ir acompañada de su precisión, empezando en 0 (la medida es errónea) y terminando en 3 (la medida está tomada a máxima precisión).

- RS4: El reloj deberá recopilar los datos proporcionados por el giroscopio. Esta medida se tomará cada cierto periodo en coordenadas X, Y y Z que expresen cuanto ha girado el dispositivo respecto a la posición anterior. La medida irá acompañada de su precisión, desde 0 (La menor precisión) hasta 3 (máxima precisión de la medida).
- RS5: Los sensores estarán configurados para que devuelvan las medidas cada cierto periodo, en función de cada sensor y de la configuración del fabricante con el sistema operativo. En Android existen los siguientes modos de recopilar las medidas tomadas por los sensores:
    - Modo de un solo disparo: Este modo devolverá la primera medida tomada. Obviamente esto no sirve si se quiere realizar un seguimiento continuo del individuo.
    - Modo continuo: Las medidas son devueltas constantemente cada cierto periodo en función de la configuración del sensor definida por el fabricante. El problema de este modo es que puede devolver muchos valores nulos por no tener nuevas mediciones en el último periodo. Por ejemplo, si no existen pasos en los 5 segundos, el sensor de pasos devolverá 0 pasos. Tras dilucidarlo, se ha decidido que este tipo de medidas son inválidas, por tener medidas poco útiles o por tener un periodo demasiado grande para algunos sensores, como el giroscopio.
    - Modo de detección de cambios: Este modo devolverá medidas cada vez que el sensor registre un cambio en las medidas tomadas. Se ha rechazado este modo puesto que en sensores como el cardíaco o el detector de pasos esto supone que si no hay variación en sus respectivas medidas (pulso o pasos por unidad de tiempo) no se notificará el estado actual de dichas medidas. Por otra parte, en sensores con mucha precisión como el acelerómetro o el giroscopio se asume un cierto margen de variación para considerar que ha habido cambio en las medidas, lo cual supone un problema en caso de movimiento o giros constantes, en donde no hay suficiente variación como para notificar el cambio.
    - Modo disparador especial: Este modo combina los mejor los dos modos anteriores. Por un lado, en sensores de gran precisión como el giroscopio, el acelerómetro o el pulsímetro devuelve continuamente mediciones aunque las medidas sean iguales o similares. Por otro lado, en sensores como el detector de pasos solo devuelve la detección de pasos cuando estos se producen. Se ha considerado que este es el modo óptimo para la actividad que se va a realizar.
- RS6: Android distingue entre la capacidad del sensor de generar medidas y la capacidad de la aplicación de recibir dichas medidas. Si el dispositivo está ocupado con muchas tareas, es posible que el sistema retrase la entrega de las medidas de los sensores a la aplicación. En determinados casos, como los juegos, esto puede ser muy perjudicial interfiriendo con la usabilidad de la aplicación, por lo que Android permite dar más o menos prioridad a la entrega de las medidas en función del retraso permitido. La aplicación debe recibir las medidas con escaso retraso, siendo más de un segundo algo perjudicial.
- RS7: Dada la variedad de dispositivos Wear, la aplicación debe estar preparada para seguir funcionando en caso de que alguno de los sensores no esté disponible en el dispositivo. Esto tiene un claro efecto negativo a la hora de dilucidar si se está teniendo una crisis o no, puesto que si falta un sensor clave como el cardíaco, podrían darse falsos positivos o falsos negativos.

- RS8: La aplicación debe ser capaz de evitar que los sensores se apaguen al ponerse la aplicación en segundo plano o cerrarse. Android Wear deja en segundo plano la aplicación al pulsar el botón inicio, al desplazar el dedo de izquierda a derecha o al tapar la pantalla. Debido a la poca potencia de estos dispositivos, al realizar cualquier acción con el reloj, la aplicación se cierra. La aplicación debe ser capaz de que los sensores sigan recopilando datos en cualquiera de estas situaciones.
- RS9: La aplicación dejará de recopilar datos biométricos de forma automática al poner los relojes a cargar o cuando la batería se encuentre en un nivel muy bajo. La razón para hacer esto no es otra que evitar la pérdida de los datos por corrupción de ficheros al cerrarse estos de forma abrupta porque se cierre la aplicación.

Así mismo, la aplicación contará con funcionalidades para el almacenamiento de las medidas anteriormente mencionadas:

- RA1: Cada medida que se vaya a almacenar debe estar acompañada de un timestamp.
- RA2: El almacenamiento de las medidas se realizará en ficheros de texto plano.
- RA3: Cada uno de los sensores contará con un fichero de texto diferente.
- RA4: Los ficheros deberán estar estructurados para su fácil visualización y procesado. Los ficheros se organizarán en columnas separadas por un tabulador. Estas columnas serán el timestamp, la precisión y el/los componentes de cada medida.
- RA5: Los ficheros contarán con dos filas a modo de cabecera de cada archivo. En la primera se detallarán los datos del sensor y el nombre del fichero. En la segunda fila, cada columna indicará el contenido de las columnas del resto del fichero.
- RA6: Dada la facilidad con que el sistema puede cerrar la aplicación, para evitar ficheros muy pesados que carguen mucho la memoria RAM del dispositivo y para disminuir las pérdidas en caso de ficheros corruptos, las medidas se irán troceando en ficheros pequeños que se juntarán más tarde fuera de este dispositivo.
- RA7: En esta aplicación de recolección, dado que no será la entregable al cliente, los ficheros no se borrarán bajo ningún concepto. Únicamente se borrarán los ficheros al desinstalar la aplicación.
- RA8: El nombre del fichero deberá contener el id del reloj, el nombre del sensor y la fecha y hora de creación del fichero.

Al no existir ninguna forma manual de acceder a los archivos guardados en el reloj, debe existir un módulo que sea capaz de transferir los ficheros al móvil, donde se pueden extraer con un explorador de archivos:

- RT1: La aplicación debe ser capaz de transmitir los ficheros guardados en el reloj sin enviarlos por duplicado.
- RT2: En caso de error al enviar un fichero, se debe reintentar. En caso de no poder en el segundo intento, este se marcará como no enviado, pero en ningún caso se borrará.

Por otra parte, como se comentó anteriormente, la aplicación está dividida en reloj o Wear (principal objetivo de este TFM) y en móvil. Para asegurarse del buen funcionamiento del protocolo de transmisión de ficheros y su almacenamiento en el móvil, se creará una

aplicación móvil de prueba que contenga dichas funcionalidades. La app móvil contará con dos módulos.

Un módulo de transmisión de ficheros con los siguientes requisitos:

- RMT1: La aplicación debe ser capaz de notificar al reloj cuándo puede enviarle los ficheros.
- RMT2: La aplicación tiene que indicarle al reloj si se ha recibido correctamente el último fichero o si ha habido algún error.
- RMT3: Para evitar que la transmisión de ficheros se haga de forma opaca al usuario, se debe mostrar una barra de progreso que indique cuántos de estos se han recibido.
- RMT4: La transmisión de ficheros entre móvil y reloj se hará necesariamente por Bluetooth, puesto que no todos los relojes poseen Wifi y el centro en el que se realizarán las pruebas, el IPP, no posee dicha instalación.

* Queda fuera de los requisitos de la aplicación la funcionalidad para que el móvil únicamente se comunique con el reloj seleccionado y no con otros con los que se hubiese emparejado. Esto se debe a que la gestión de emparejamiento, conexión y selección del reloj solo se puede hacer desde la aplicación Android Wear.

Por último, un módulo encargado del almacenamiento de ficheros:

- RF1: La aplicación tiene que ser capaz de almacenar los ficheros en una carpeta accesible desde un explorador de archivos para móviles.
- RF2: La aplicación debe ser capaz de recoger los ficheros recibidos y clasificarlos quedando repartidos en una carpeta para cada reloj y un fichero para cada tipo de medidas. Para su ordenación, se hará uso de la fecha y hora del nombre del fichero.
- RF3: Los ficheros recibidos deben ser borrados una vez se haya terminado la mezcla.
- RF4: Al realizar la mezcla de ficheros, la aplicación debe hacerlo en orden cronológico de las medidas.

4.2.2 - Requisitos no funcionales

Por otra parte, se ha puesto como objetivo los siguientes requisitos no funcionales, comunes a ambas aplicaciones:

- RNF1: Interfaz simple. Teniendo en cuenta las circunstancias de los usuarios, la aplicación debe tener una interfaz fácil de utilizar. En el reloj se debe mostrar únicamente el botón de start-stop para iniciar o detener las mediciones. En el caso del móvil se mostrará el botón para iniciar la recogida de datos.
- RNF2: Cumplimiento de la ley de protección de datos (Estado, 1999). La aplicación no debe enviar los datos personales a ningún servidor. Los datos biométricos recogidos para el estudio estarán siempre en los dispositivos de desarrollo de la aplicación.

- RNF3: Datos anonimizados. Todos los datos recopilados estarán anonimizados y no debe existir forma alguna de relacionarlos con los usuarios (Anonimización, 2014).
- RNF4: Previsión ante acciones no intencionadas. Debido a que la aplicación del reloj será utilizada por niños, la aplicación debe seguir funcionando con normalidad pese a agitaciones del reloj, tapar la pantalla o desplazar el dedo de izquierda a derecha. Estas acciones cierran las aplicaciones o apagan la pantalla. También es importante que la aplicación deje de recopilar datos o detenga la transmisión de ficheros en el momento en el que se detecte que el reloj está cargándose o con poca batería. Esto último es aplicable también para el móvil.
- RNF5: Diseño *responsive*. La aplicación *mobile* debe ser usable tanto en móviles como en tabletas. La aplicación Wear debe funcionar en relojes cuadrados y circulares.

4.3 - Diseño de la aplicación

4.3.1 - Diseño del sistema

Tal y como se ha indicado en los requisitos, el sistema se compone de dos aplicaciones Android que deben estar conectadas entre sí: Una aplicación *mobile* y otra para *smartwatch* (Android Wear). La aplicación para Android Wear será la encargada de recopilar los datos biométricos de los usuarios y de almacenarlos hasta que el otro dispositivo, el móvil, los recoja. La aplicación móvil debe ser la encargada de recoger los datos de los relojes y de ordenarlos en ficheros por sensor, de forma cronológica y siempre separándolos en una carpeta para cada reloj.

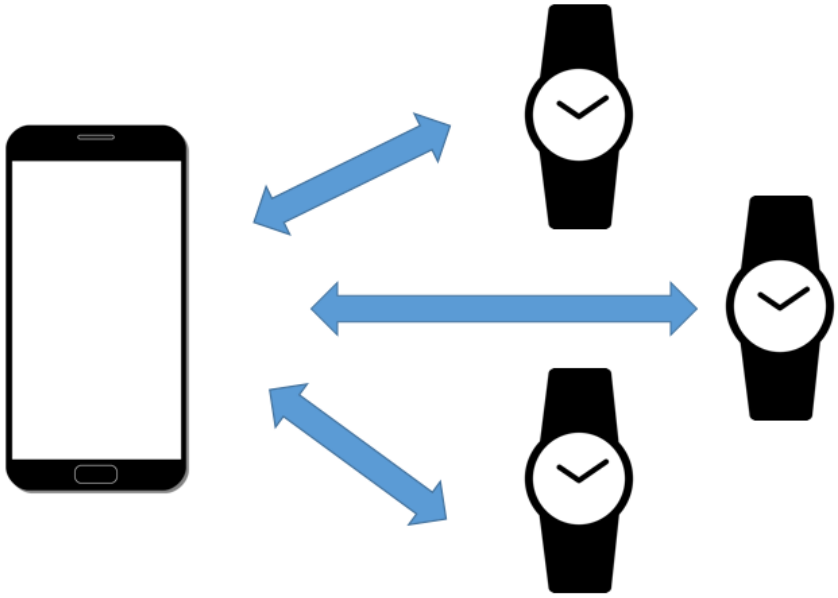

Ilustración 5: Conexión entre móvil y relojes

Como se puede deducir de los requisitos, el móvil actúa en parte como si fuese un servidor, puesto que es el que almacena los datos y los ordena tras recibirlos y el que contiene todos los datos de todos los relojes. Por otra parte, como se ha comentado, los ficheros deben estar en una carpeta accesible por un explorador de archivos, bien a través del ordenador o a través de una app que tenga esta funcionalidad, para poder recogerlos, procesarlos y así obtener el perfil que indique si se está produciendo una crisis o no.

Aplicación Android Wear

La aplicación para relojes es la encargada de recopilar y almacenar datos biométricos y de enviarlos cuando el móvil con el que está emparejado el reloj lo requiera. Estas tres funcionalidades serán los tres subsistemas que formen parte de la aplicación.

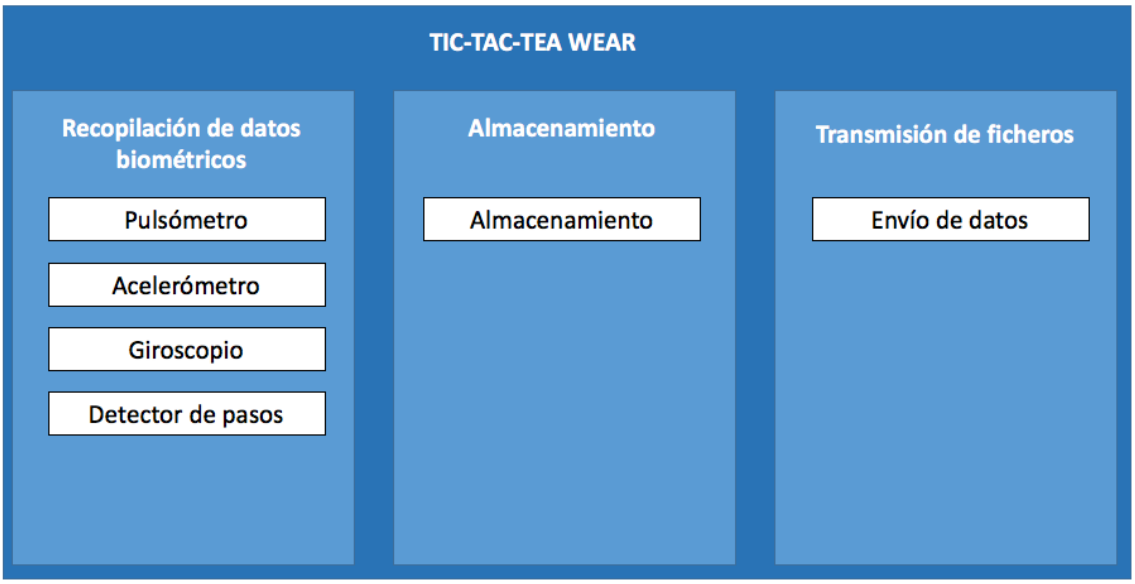

Ilustración 6: Módulos de TIC-TAC-TEA Wear

- Subsistema de recopilación de datos biométricos: Es el encargado de manejar cada uno de los sensores del dispositivo y la recolección de los datos que estos arrojan.

    - Módulo del sensor cardíaco. Es el encargado de recopilar los datos del sensor cardíaco. Este sensor medirá cada cierto tiempo las pulsaciones detectadas desde la medición anterior, o desde el encendido del sensor si se trata de la primera medición, y establecerá la media por minuto en cada medición. Por ejemplo, en caso de detectar en un periodo de 5 segundos 10 pulsaciones, la media de pulsaciones por minuto sería de 120 pulsaciones/minuto. Las medidas por minuto son las que se devuelven a la aplicación. Por otra parte, las medidas irán acompañadas de un *timestamp* y un número que indica la precisión de la medida tomada (siendo 0 la menor precisión y 3 la máxima precisión).

    - Módulo del sensor de pasos. Este módulo es el encargado de recoger los movimientos detectados como un paso cada vez que se produzcan. A diferencia del sensor anterior, en este se recoge el *timestamp* y la precisión únicamente cuando se produce este evento. En este caso la precisión de 0 a 3 corresponde a la similitud del movimiento del dispositivo al balanceo de los brazos en un paso.

    - Módulo del acelerómetro. Es el módulo que gestiona la recolección de los datos arrojados por el acelerómetro. Este sensor devuelve mediciones cada cierto tiempo de la aceleración del dispositivo. La configuración de la frecuencia con la que devuelve medidas y la sensibilidad mínima en la variación de la aceleración para considerarse movimiento viene dado por el fabricante del sensor y el fabricante del reloj. Las medidas vendrán en sus componentes X, Y y Z y estarán acompañadas de la precisión de la medida realizada y del tiempo. La precisión se moverá en el intervalo de números enteros desde 0, la menor precisión y posiblemente una medida errónea, a 3, la máxima precisión.

- Módulo del giroscopio. Es el módulo que gestiona la recolección de los datos devueltos por el sensor del giroscopio. Este sensor devuelve mediciones cada cierto tiempo cuando existe una variación suficiente de movimiento giratorio tomando como punto de referencia a la hora de medir giros el centro del sensor ubicado en el reloj. La configuración de la frecuencia con la que devuelve medidas y la sensibilidad mínima para considerarse movimiento viene dado una vez más por el fabricante del sensor y el del reloj. Las medidas vendrán en sus componentes X, Y y Z y vendrán siempre con una medida de la precisión de la medida realizada (de 0 a 3, siendo 0 la menor precisión y 3 el máximo) y del timestamp.

- Subsistema de almacenamiento. Es el encargado de guardar las medidas de los sensores en ficheros de texto. Cuenta con un único módulo:

    - Módulo de almacenamiento. Guarda las medidas de los sensores en ficheros de texto. Cada sensor contará con sus propios ficheros. Por seguridad y, como se explicará más adelante, por eficiencia a la hora de enviar datos al reloj, no se guardarán todos los datos de cada sensor en un solo fichero. Se guardarán en ficheros que almacenen un determinado periodo de tiempo.

- Subsistema de transmisión. Este es el subsistema encargado de transferir archivos desde el reloj al móvil al que está sincronizado.

    1. Módulo de envío de datos. Es el que, al recibir la orden desde el móvil, envía todos los ficheros almacenados hasta el momento al móvil. Por cada fichero enviado se recibirá una confirmación desde el móvil para pasar al siguiente archivo. Los ficheros que hayan sido enviados correctamente o no, serán marcados para no ser enviados de nuevo. Nunca se eliminarán estos ficheros hasta que no se instale la segunda aplicación desarrollada en este TFM.

Aplicación Android Mobile

Debido a que el principal objetivo de este TFM es la aplicación para Android Wear, la aplicación para Android *mobile* únicamente servirá para la recolección de los datos almacenados en el reloj, puesto que estos son inaccesibles de forma directa en el reloj. Una vez se reciban los ficheros, estos se mezclarán separándose de forma que solo haya un fichero por sensor. Cada reloj, en caso de haber más de uno, estará también separado en carpetas y tendrá cada uno sus propios archivos.

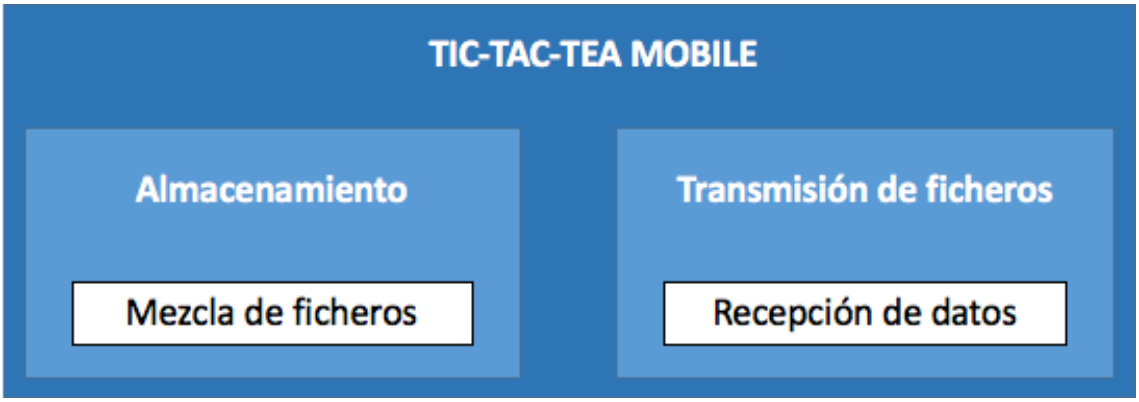

Ilustración 7: Módulos de TIC-TAC-TEA mobile

- Subsistema de recepción. Este subsistema es el que tiene como objetivo recibir todos los ficheros que tiene el reloj con el que está sincronizado.

    1. Módulo de recepción de ficheros. Una vez el usuario indique a la aplicación del móvil, a través de un botón situado en la pantalla, que se quiere recibir los ficheros, el móvil comunicará al reloj sincronizado que debe enviar todos los ficheros que no haya enviado con anterioridad. Los archivos recibidos se guardarán directamente desde este módulo, puesto que lo que se recibe es el fichero en sí.

- Subsistema de almacenamiento. Es el encargado de almacenar los ficheros recibidos. Cuenta con un único módulo.

    1. Módulo de mezcla de ficheros. Puesto que los ficheros se guardarán directamente desde el subsistema anterior, este subsistema solo contará con este módulo que se encargará de mezclar los ficheros por sensor en orden cronológico y separando los ficheros de los distintos relojes.

Comunicación Mobile-Wear

La aplicación de TIC-TAC-TEA está dividida en dos partes: La aplicación para móvil y la aplicación para reloj. Debido a las limitaciones del sistema operativo del reloj, Android Wear, los archivos que guarde la aplicación no pueden ser accedidos mediante un explorador de archivos, por lo que deben ser transferidos al móvil desde la propia aplicación de TIC-TAC-TEA. La transmisión de los datos se hará necesariamente mediante Bluetooth, puesto que es una característica que tienen en común todos los relojes de Android Wear y todos los móviles.

En un principio, se ha pensado que el flujo de datos entre ambas aplicaciones sea el que se muestra en la siguiente figura.

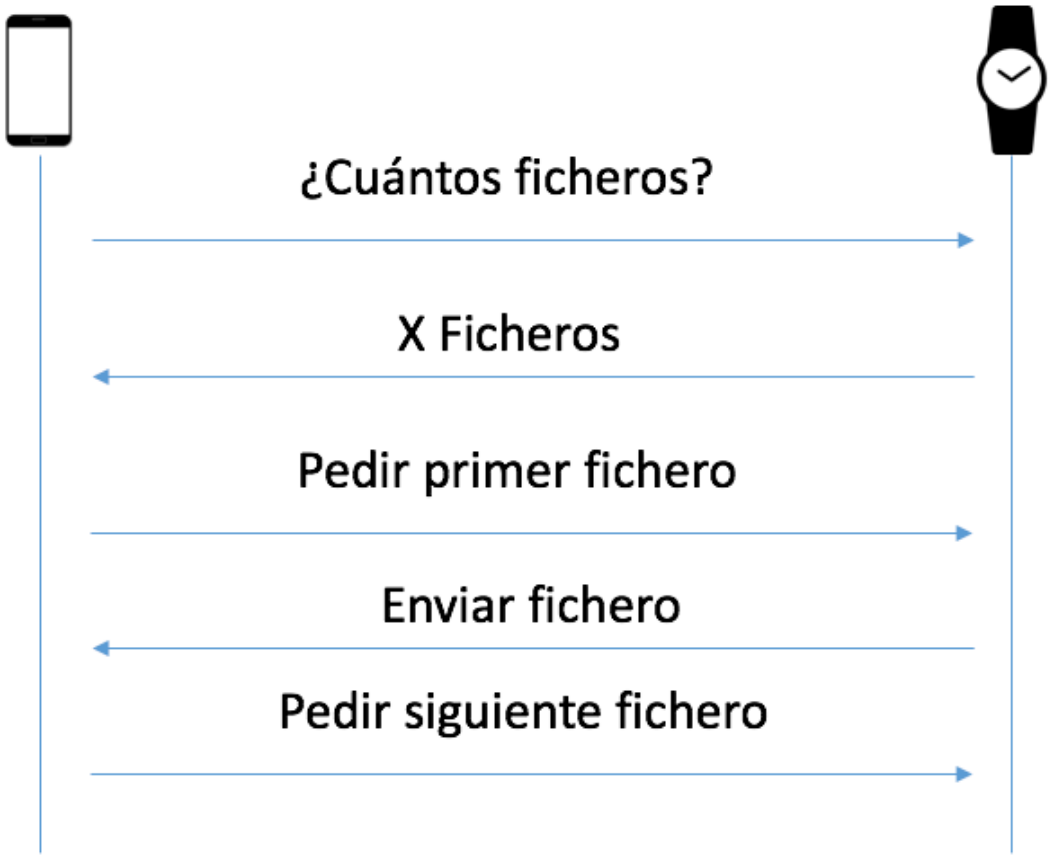

Ilustración 8: Diagrama de flujo de datos

Los pasos que deberá seguir este protocolo son los siguientes:

1. El móvil pregunta al reloj cuántos ficheros tiene para transferir.
2. El reloj responde con el número pedido. Este número se utilizará en el móvil para indicar al usuario de forma visual el progreso de la transmisión de los datos.
3. El móvil indica al reloj que inicie la transmisión de los ficheros que tenga.
4. El reloj envía el primer archivo no enviado.
5. El móvil indica al reloj que envíe el siguiente.

Los pasos 4 y 5 se repetirán hasta que se termine la transferencia de los datos.

4.3.2 - Diseño gráfico

Dado que esta primera aplicación está orientada a la recopilación de datos, el diseño gráfico solo se aplicará a lo imprescindible para que esa funcionalidad esté activa.

Aplicación Wear

La aplicación para los relojes Android Wear será la que tengan que utilizar los profesores del Instituto de Psicopediatría en las pruebas con sus alumnos para recopilar datos durante varias semanas. Por este motivo, el diseño de la aplicación debe ser muy simple y sencillo de utilizar.

Ilustración 9: Plantilla de la actividad principal

Como se ve en la figura anterior, la aplicación cuenta únicamente con un único botón circular con el texto *"Start"* para iniciar las mediciones o *"Stop"* para detenerlas.

Aplicación Mobile

La aplicación para móviles Android será utilizada únicamente por el personal del laboratorio Amilab para la recolección de los datos guardados en todos los relojes que estarán recopilando datos en el IPP.

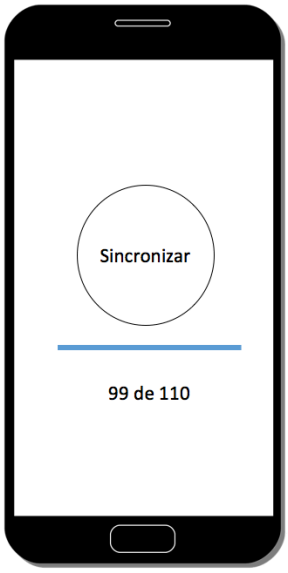

Ilustración 10: Plantilla de la actividad principal

Como se aprecia en esta figura, la aplicación dispone de tres elementos: Un botón para iniciar la transferencia de los archivos guardados en el reloj, una barra de progreso que indique el estado de la transferencia y un indicador que muestre los archivos recibidos sobre el total. Aunque la aplicación vaya a ser utilizada solo por el personal del Amilab, es importante indicarle el progreso de la transferencia de los archivos para no cortar la conexión móvil-reloj antes de tiempo o esperar más de lo necesario.

4.3.3 - Interacción con el usuario.

En esta sección se detallará la interacción de los diferentes usuarios con la aplicación, tanto en su versión para móvil como para reloj.

Aplicación Wear

Como se ha podido ver en la sección anterior, la aplicación para reloj contará solo con una única vista, que es la que inicia o termina con las mediciones.

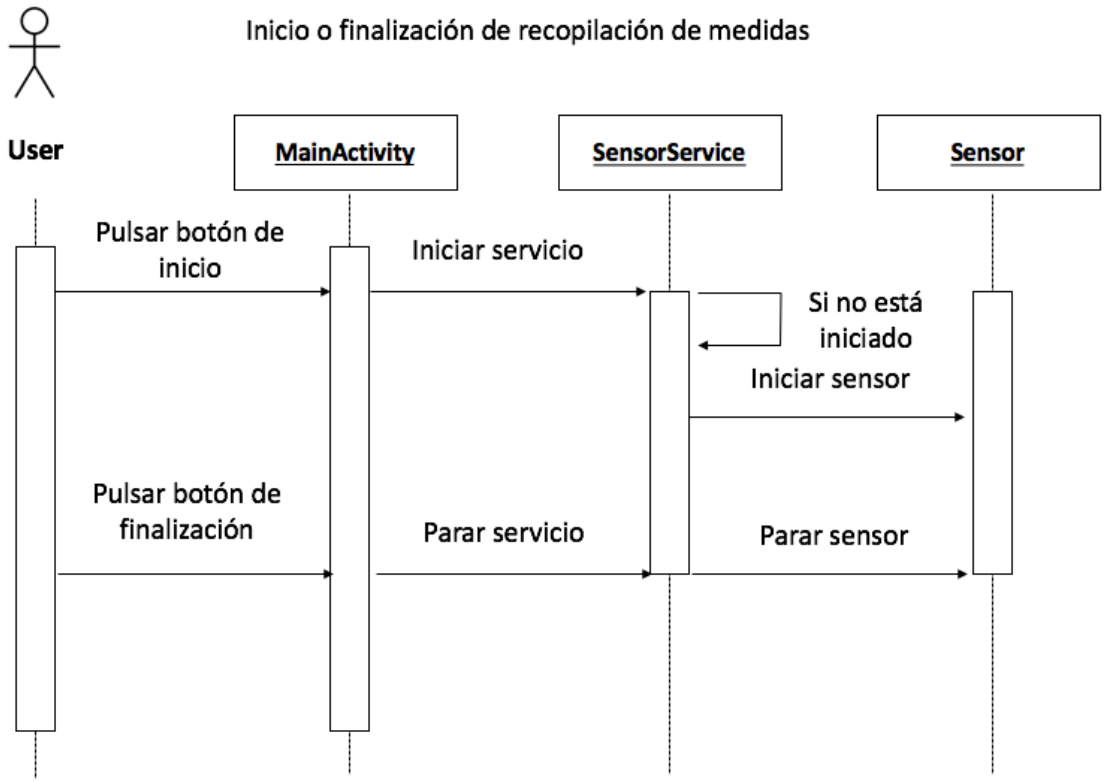

Ilustración 11: Diagrama de interacción para inicio/detección de las mediciones

Puesto que la aplicación para reloj será iniciada por los profesores el IPP, este proceso debe ser lo más simple posible. Únicamente se habrá de pulsar el botón Start para comenzar con las mediciones.

Por otra parte, la detención de las mediciones debe tener el punto justo de facilidad para que el personal del IPP pueda detenerlo pero que a la vez sea muy difícil para un niño detenerlo de forma accidental. En esta ocasión se existen dos formas de parar la recolección de datos. Una de ellas es pulsando durante 10 segundos el botón Stop y otra es poniendo el reloj a cargar. La aplicación debe seguir recogiendo mediciones con cualquier otro gesto que cierre la aplicación, como puede ser desplazar el dedo de izquierda a derecha, tapar la pantalla con la mano o agitar el reloj de una determinada forma. Se detallará cómo se ha realizado esto en el apartado de implementación.

Aplicación Mobile

En el caso de la aplicación para móviles, la interacción con el usuario varía poco respecto a la versión para relojes. En este caso, el usuario deberá pulsar únicamente el botón de sincronizar para que el móvil empiece a recoger los ficheros del reloj actualmente sincronizado. Una vez pulsado, se hará visible la barra de progreso que muestre los archivos recibidos y, tras la mezcla de los ficheros, estarán disponibles para su recogida a través de una aplicación que provea de un explorador de archivos.

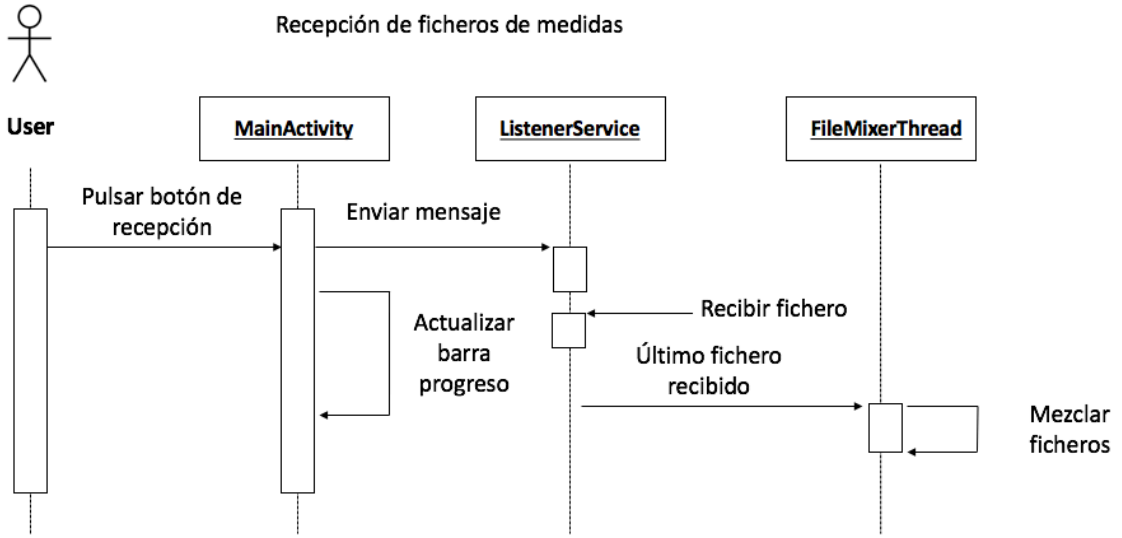

Ilustración 12: Diagrama de interacción para recibir ficheros

4.4 - Desarrollo de la aplicación

4.4.1 - Introducción

En este apartado se va a explicar cuáles han sido las pautas seguidas para la implementación del sistema TIC-TAC-TEA. Para cada una de las dos partes se explicará qué tecnologías se han utilizado, que decisiones se han tomado y se mostrará el resultado obtenido.

4.4.2 - La aplicación Wear

La aplicación para reloj ha sido desarrollada enteramente en Java, bajo el SDK de Android Wear en su versión 4.4 (o Wear 1.0). El motivo de utilizar esta plataforma es que ofrece una mayor variedad de dispositivos, con diferentes sensores y un intervalo más amplio de precios que la principal competencia, Apple Watch. Por otra parte, la versión utilizada es la mínima que tendrán todos los relojes de Android Wear, por lo que no hay que preocuparse de actualizar el reloj a la última o de si este es compatible o no con dicha versión. Utilizando Android Wear 1.0 se tiene acceso al mercado completo de relojes Android.

El patrón Modelo Vista Controlador

A la hora de diseñar los módulos de esta herramienta se ha utilizado la arquitectura del Modelo Vista Controlador. Se ha separado entre grupos diferentes la interfaz gráfica, la lógica que recoge los datos e interacciones del usuario con la interfaz y la lógica de negocio y la parte de persistencia de datos.

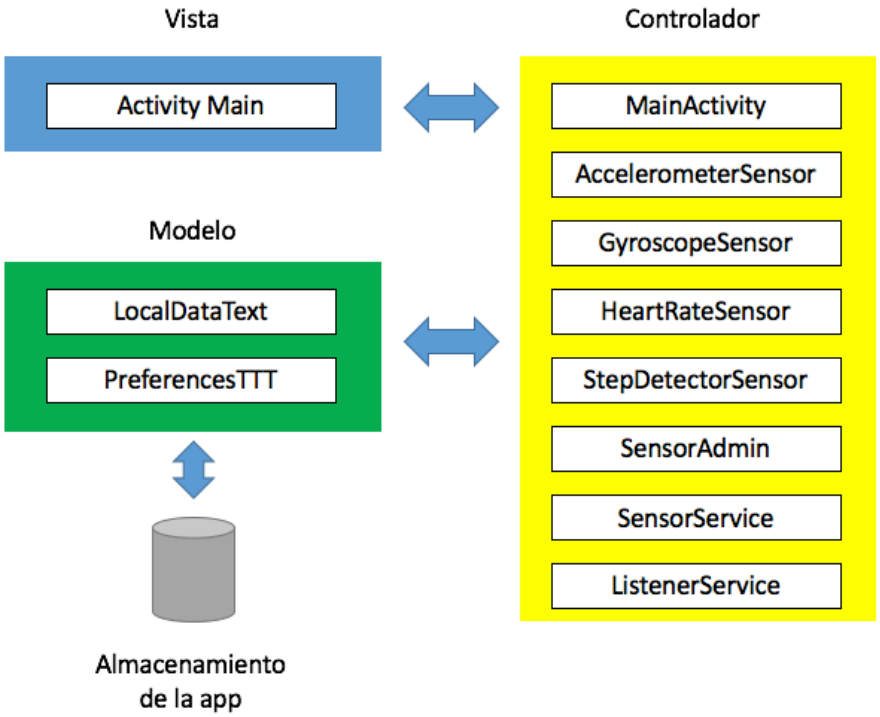

Ilustración 13: Modelo-Vista-Controlador de TIC-TAC-TEA Wear

1. Vista: Es la interfaz gráfica de la aplicación Android, la parte con la que los profesores interactuarán para iniciar o detener la recogida de los datos biométricos. La interfaz está hecha utilizando ficheros XML y utilizando los elementos que pone a disposición del desarrollador el SDK de Android.

2. Modelo: Es la capa de persistencia de datos. En este caso concreto se trata de las clases que organizan y guardan las medidas de los sensores en ficheros de texto. Estas clases están implementadas en Java, utilizando el API de Android. Por otra parte, la clase de PreferencesTTT es la que almacena los pares clave-valor que contienen el estado en el que se encuentra la aplicación, para así poder continuar en el punto en el que estaba en caso de algún tipo de interrupción. Por ejemplo, en caso de que el usuario cierre la aplicación mientras se están realizando mediciones, existe una preferencia con la que se impide que se vuelva a crear otra escucha de los sensores.

3. Controlador: Es por una parte el código que recoge las interacciones y datos que el usuario introduce a través de la interfaz gráfica y la lógica interna de la aplicación que puede modificar además la interfaz. Esta implementado en clases Java bajo el API de Android SDK. La clase MainActivity.java es el controlador de la vista con el mismo nombre, pues recoge las interacciones del usuario, las pulsaciones del reloj y modifica el texto del botón en función del estado de las mediciones.

Antes de describir el sistema más en detalle, se comentarán los principales puntos del desarrollo de la aplicación, para dar una visión más amplia de las decisiones tomadas.

El problema de los sensores

Como se ha comentado anteriormente, uno de los requisitos para esta aplicación es que los sensores funcionen siempre, incluso si la aplicación es cerrada. Para poder escuchar de forma continua a los sensores, es necesario que haya un proceso recibiendo continuamente las medidas.

El API de los eventos de los sensores, aquella que maneja las medidas recibidas, funciona en un hilo diferente. Siempre que se llame a las funciones que manejan los cambios en las medidas, las cuales son las funciones que hay que implementar siguiendo las especificaciones de la interfaz *SensorEventListener,* se hará en un hilo diferente que muere tras terminar la función y al cual no se tiene acceso desde otro hilo.

Un reto que se encontró al poco de empezar la aplicación fue el de la duplicidad de las medidas. La primera implementación de los sensores registraba la escucha de los eventos de los sensores al iniciar la primera actividad. Puesto que un requisito de la aplicación es la medición continua de las mediciones, se optó por no terminar las escuchas al cambiar de actividad (o cerrar la aplicación, lo que implica cambiar la actividad). El problema de esta implementación es que al volver a abrir la aplicación se volvía a registrar una nueva escucha mientras la anterior seguía funcionando, por lo que las medidas se duplicaban al existir dos escuchas.

Un intento de arreglar el problema anterior fue el de guardar en una preferencia de la aplicación, la cual es un modo de persistencia de datos basados en clave/valor, que indicase si ya existía una escucha activa. Sin embargo, aunque con esto se evitaba volver a crear escuchas, surgía un nuevo problema: cómo parar manualmente la recolección de datos. Al volver a arrancar la aplicación, la instancia para iniciar o detener las escuchas es distinta en cada ocasión, provocando que no se pudiese parar las escuchas que estuviesen activas por mucho que se supiese el estado de los sensores con el valor guardado en las preferencias.

Otro intento de solucionar el problema fue la creación de una tarea asíncrona, el cuál es parte de la API de Android, pero una vez más el problema era la instancia de la aplicación. Las tareas asíncronas están pensadas para detenerse ellas mismas puesto que funcionan independientemente del resto de procesos. Al no poder acceder de ninguna forma a esta tarea tras reabrir la aplicación, una vez más por la instancia de la aplicación, el hilo quedaba funcionando eternamente hasta que el sistema lo cerrase.

La solución a esta incidencia fue la creación de un servicio. Los servicios son hilos o procesos (según se configure) secundarios independientes de la instancia de la aplicación y son únicos. Solo puede existir un servicio de la misma clase, la cuál hereda de la clase padre *service*. Otra ventaja de usar servicios es que si se intenta crear un nuevo hilo de la misma clase, automáticamente se devuelve el servicio ya creado y en caso de querer destruirlo solo hace falta indicar su clase. Esto permite arrancar tantas veces como se quiera la aplicación sin que se dupliquen las medidas y parar las mediciones pidiendo al sistema que destruya el único servicio existente de este tipo.

Almacenamiento de datos

Tal y como se comentó en anteriores secciones, el objetivo principal de esta primera aplicación es el de la recolección de datos biométricos. Estos datos han de guardarse en un lugar en el que no se pierdan y al que se pueda acceder de alguna forma. Tras valorarlo, se ha elegido como mejor opción la de guardar los datos en ficheros de texto plano y no en una base de datos puesto que se transformarán en ficheros igualmente.

Como los datos serán analizados para obtener el perfil de estrés que será introducido en la aplicación final, estos deberán tener una estructura fácilmente legible, independientemente de a qué sensor corresponda el fichero. Se ha optado por añadir a cada fichero una cabecera que indique los datos del sensor y utilizar una línea para cada medida. Estas medidas estarán formadas por el timestamp de la medida, la precisión de esta y el/los componentes que forman la propia medida. Las medidas como los pasos o el pulso cardíaco solo contarán con una componente mientras que otras medidas como las del acelerómetro o el giroscopio tendrán tres componentes.

Entrando en detalles más técnicos, se ha podido apreciar que no es factible acceder decenas de veces a disco por segundo para abrir un fichero y colocar al final de este la nueva medida correspondiente. Para evitar esto se ha hecho uso de un buffer en el que se guardarán las medidas que se van obteniendo hasta que llegue el momento del volcado a disco. Cada uno de los sensores cuenta con su propio buffer para que, cada cierto tiempo, se vuelquen los datos en un fichero de texto nuevo correspondiente al sensor. Hay que subrayar en la expresión “fichero nuevo”, puesto que por seguridad se ha optado por guardar cada volcado en un fichero distinto al anterior, para disminuir la pérdida de datos

en caso de corrupción de ficheros. Por otra parte, como se va a explicar más adelante, la única opción disponible para transmitir datos es el Bluetooth, pese a tener dificultades para transmitir ficheros grandes, lo cual es otra razón más para reducir el tamaño de los ficheros.

Estos ficheros se transmitirán al móvil donde serán mezclados en un fichero por cada sensor, ordenando las medidas cronológicamente.

Estructura del sistema

Como se ha detallado en el apartado de diseño, la aplicación para Android Wear está dividida en tres subsistemas:

1. Subsistema de recopilación de datos biométricos: Es la encargada de recopilar los datos biométricos de los cuatro sensores del reloj: El pulsómetro, el giroscopio, el acelerómetro y el detector de pasos. La clase que inicia todo es la clase *SensorService*, el cuál es un servicio iniciado desde el momento que el usuario pulsa el botón Start en la pantalla de inicio. Como se explicó anteriormente, puesto que los servicios no se duplican, el inicio de las mediciones se realiza cuando se crea este servicio por primera vez. En caso de que ya estuviese creado, el servicio mantiene en memoria un booleano activo con el que se rechaza la creación de nuevo de los sensores, evitando así la duplicidad de los sensores. Esta clase implementa también el *listener* que recibe las medidas devueltas por los sensores en forma de evento. Para no mezclar el servicio en sí y el manejo de los sensores, se ha implementado la clase *SensorAdmin*, la cual se encarga de procesar esas medidas en función del sensor que las haya recopilado y guardarlas en el buffer correspondiente. También es el que indica al sistema operativo qué sensores se van a escuchar al iniciarse el servicio y cuándo se deben detener estas escuchas.

   Cada uno de los sensores tiene su propia clase, con un modelo de los datos único para ese sensor (medidas en sus componentes, precisión y *timestamp*), y una instancia propia de la clase que escribe en fichero, LocalDataText.

2. Subsistema de almacenamiento de medidas: Es el encargado de almacenar las medidas recibidas en ficheros de texto. La clase anteriormente mencionada, *LocalDataText*, almacena las medidas que va recibiendo del sensor en un buffer en formato *String[]* y, tras 15 segundos, lo vuelca en el archivo de texto. También es la clase que cierra los ficheros cada 10 minutos, tiempo máximo estimado en el que los ficheros son suficientemente pequeños para ser enviados al móvil sin problemas. Además, en caso de que el servicio se cierre, bien por petición del usuario o por petición del sistema operativo, se comunicará a esta clase que vuelque inmediatamente los buffers en sus correspondientes ficheros de texto y los cierre de forma segura. De esta forma, se reduce la probabilidad de que haya problemas con los ficheros por no cerrarse correctamente.

3. Subsistema de transmisión: Es el que tiene la funcionalidad de enviar los ficheros de texto no enviados anteriormente tras recibir una petición desde la aplicación alojada en el móvil. Está formado por una única clase: *ListenerService*, llamada así porque su función durante la mayor parte del ciclo de vida de la app es escuchar las peticiones del móvil. Al heredar de la clase *WearableListenerService* e

implementar el *GoogleApiClient*, esta clase recibirá todas las peticiones dirigidas a esta aplicación.

A modo de filtro, todos los paquetes de datos recibidos y los mensajes vienen acompañados de un *path* utilizado para indicar qué aplicación ha enviado el mensaje o a modo de comando. En este caso, se ha optado por utilizar cuatro comandos personalizados diferentes:

- Uno para pedir al reloj que envíe un mensaje con el número de ficheros que se van a transmitir. Con este número el móvil podrá indicar al usuario cuántos ficheros faltan por enviar.
- Un comando para enviar el primer fichero disponible.
- Un comando para indicar que se ha recibido un archivo correctamente. Al recibir este comando, se marcará el fichero indicado en el mensaje como 'enviado' y se enviará el siguiente archivo que esté disponible. En caso de no haber más ficheros disponibles, se mandará un mensaje indicando que se ha finalizado la transmisión.
- Un comando que indica que se envíe el siguiente fichero. Este comando es similar al anterior, pero indica que ha habido algún problema con los que se estaba enviando. No se incluye el nombre del fichero enviado, puesto que cabe la posibilidad de que ni siquiera el nombre del fichero enviado sea correcto. En este caso, se seleccionará el primer fichero disponible para enviar, el cuál es el que ha dado problemas, y se marcará como problemático, dejándolo por si se puede recuperar en el futuro. A continuación se cogerá el siguiente fichero disponible. En caso de no haber más ficheros disponibles, se mandará un mensaje indicando que se ha finalizado la transmisión.

  La mayoría de los ficheros que no han podido enviarse correctamente, como se pudo comprobar más adelante, no fue por problemas de corrupción de archivos, fue porque estos sobrepasaban el tamaño en memoria que admitía la cola de envío del reloj, lo cual hacía que el móvil recibiese un archivo incompleto. Utilizando la implementación mejorada del módulo de transmisión de la aplicación de regulación emocional (la aplicación de la segunda fase), se pudo recuperar estos archivos.

Por último, de forma transversal a los tres módulos, existe una clase, llamada *BootStartService*, que ha sido registrada para arrancar con el inicio del sistema. A través del fichero de manifiesto de las aplicaciones Android, esta clase se puede iniciar manualmente desde la aplicación o por el sistema operativo al terminar el arranque. Por su parte, esta clase implementa *BroadcastReceiver*, con lo que es capaz de recibir eventos del sistema o eventos personalizados lanzados por la aplicación. El objetivo de la clase es detectar cuando se pone a cargar el reloj o tiene poca batería para detener el servicio de las mediciones y que se cierren los archivos correctamente.

Otra clase que funciona de manera transversal es una de las encargadas de la persistencia de datos, *PreferencesTTT*, que hace uso del fichero de preferencias que cada aplicación puede disponer. Los ficheros de preferencias almacenan datos en forma de parejas de clave/valor, y en este caso se ha utilizado para guardar si la aplicación está recopilando medidas o no. Esta preferencia es utilizada principalmente por el botón de la pantalla de la aplicación, para saber si debe mostrar el texto "Start" o "Stop".

El problema de la identificación de los relojes

Una de las dificultades encontradas a la hora de desarrollar la aplicación ha sido cómo realizar la identificación de los relojes. La aplicación "Android Wear" está pensada principalmente para tener un único móvil/Tablet y un solo reloj emparejado con él. Sin embargo, en la versión Android se permite, no sin cierta dificultad, administrar varios relojes al mismo tiempo. Sabiendo eso, la aplicación TIC-TAC-TEA para móvil debe ser capaz de administrar los datos de varios relojes al mismo tiempo, aunque el diseño de estos dispositivos esté pensado para ser usado en parejas.

En este caso concreto, la aplicación de recopilación tenía que funcionar solamente con un móvil que recoja los datos de hasta 11 relojes que se encuentran disponibles para el proyecto. Esta situación ha traído un problema, ¿Cómo identificar los relojes? Por una parte, resulta imposible introducir un nombre o un apodo en el reloj, puesto que no existe teclado (pendiente de la próxima versión de Android Wear). Por otra, no es posible relacionar los datos con una persona, por la ley de protección de datos, únicamente se sabe que el reloj X tiene unas medidas recopiladas y se tienen anotaciones de que la persona que llevase ese reloj en ese momento ha tenido crisis en las horas anotadas.

Lo primero que se ha descartado es utilizar el número de serie, puesto que la mayoría de los relojes utilizados en la recopilación de datos tienen el mismo número de serie.

El segundo intento fue utilizar la MAC de los relojes, puesto que es única en cada dispositivo. Aunque los relojes tienen cerrada la conexión a internet a través de software, estos pueden conectarse a redes Wifi para descargar las actualizaciones. Sin embargo, la MAC fue finalmente descartada porque los relojes debían estar conectados a la red Wifi para obtener dicho número, una vez más por limitaciones del SDK. Esto resultaba un problema importante porque el centro donde se recogen las medidas no disponían de Wifi en el momento de hacer las pruebas.

Por último, tras recorrer el API de Android, se encontró un método que devolvía un identificador con baja probabilidad de repetirse. Hay que subrayar el concepto de "baja probabilidad de repetirse", puesto que el identificador se genera semi-aleatoriamente al configurar por primera vez el reloj. Además este número se vuelve a generar al restaurar el dispositivo a su estado de fábrica.

Con este número, visible también desde el menú de opciones del propio reloj, se pudo identificar a cada uno de los relojes y mantener separadas sus medidas.

4.4.3 - La aplicación Mobile

Esta aplicación, pese a no ser la definitiva de TIC-TAC-TEA cumple con la funcionalidad imprescindible de recoger las mediciones de los relojes, cosa que no se puede hacer de otra forma como utilizando un explorador de archivos.

Puesto que la aplicación para reloj se ha realizado bajo la plataforma Android Wear, la aplicación emparejada debe desarrollarse bajo Android también, puesto que en el momento de su desarrollo no existía forma de conectar una aplicación iOS con una Android Wear.

El patrón Modelo Vista Controlador

Al igual que con la aplicación para el reloj, se ha seguido también el patrón del Modelo Vista Controlador en el diseño de la aplicación para móvil. Las clases utilizadas se han dividido en los tres grupos: La vista para la interfaz gráfica, el modelo para la persistencia de datos y en el controlador la lógica y la interacción con la vista.

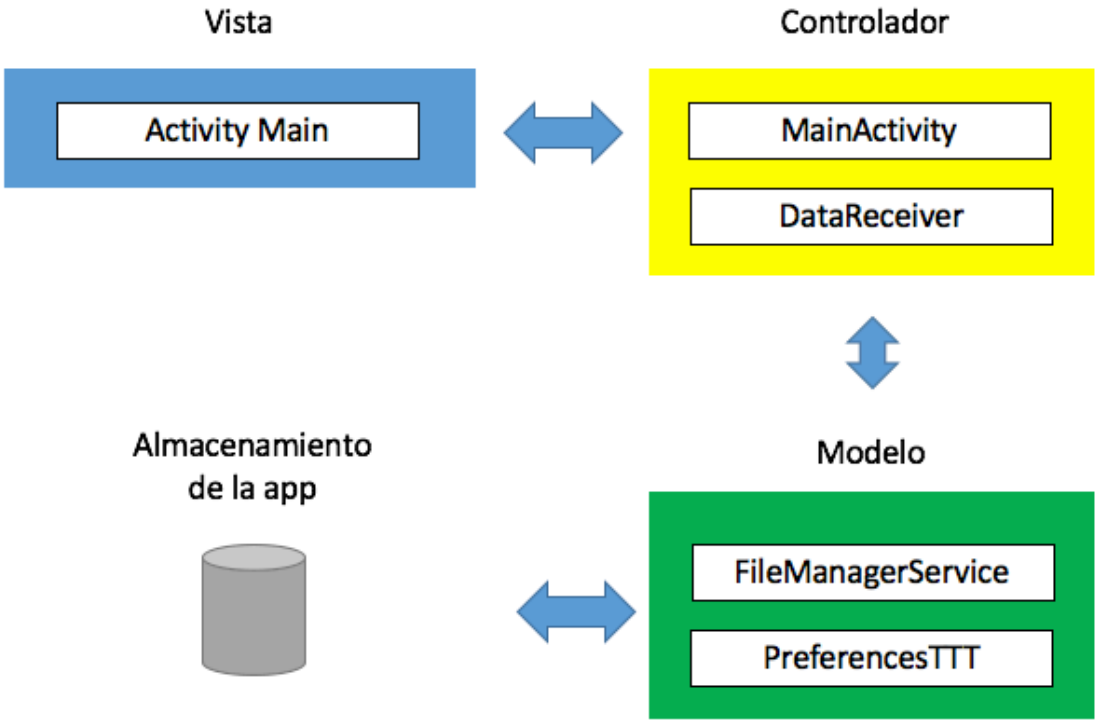

Ilustración 14: Modelo-Vista-Controlador de TIC-TAC-TEA Mobile

1. Vista: Es la interfaz gráfica de la aplicación móvil, construida con los elementos que provee el SDK de Android para los ficheros XML que describen la interfaz. El personal del Amilab será el que interactúe con esta interfaz para poder recoger los ficheros de los relojes.

2. Modelo: Es la capa de persistencia de datos. En el caso de la aplicación para móvil se trata de la clase que se encargará de la mezcla y guardado de los ficheros recibidos. Se incluye también en esta categoría la clase de preferencias. Todas las clases está implementadas en Java utilizando el API de Android.

3. Controlador: Es la parte del código que contiene la lógica de la aplicación, la encargada de recoger las interacciones del usuario en la interfaz y la que modifica dicha interfaz si la lógica lo requiere. Está implementado en clases Java bajo el API de Android SDK.

Estructura del sistema

Tal y como se ha explicado anteriormente, en el apartado de diseño, la aplicación para móvil esta formado por dos módulos principales:

1. Recepción: Es el módulo encargado de la recepción de los ficheros en el móvil. Desde la interfaz de usuario, en la clase de *MainActivity*, tras pulsar el botón de sincronizar se envía un mensaje al reloj reclamando los ficheros de las medidas que no hayan sido enviados. Una vez el reloj comience a enviar los ficheros, la clase *DataReceiver*, que hereda de *WearableListener* e implementa las clases *GoogleApiClient*, recibirá todos los paquetes de datos que le envíe el reloj y, si el *path* que acompaña a todo paquete contiene uno de los comandos reconocidos por la aplicación, se descargará el archivo correspondiente.

2. Almacenamiento: Es el módulo encargado de guardar los ficheros. Cuenta con un servicio llamado *FileManagerService* que, tras recibir los ficheros, mezclará en segundo plano los ficheros por orden cronológico de sus medidas en ficheros dedicados a cada sensor y cada reloj.

La mezcla de ficheros

En esta memoria se ha dedicado un apartado extra a la mezcla de ficheros debido a que su implementación no es simple.

Por una parte, como se ha visto en el apartado anterior, la mezcla de ficheros se ha llevado a cabo en un servicio. Al ser una tarea pesada no es posible ejecutarla en el hilo principal puesto que se bloquearía la aplicación y el sistema operativo terminaría con ella. En este punto existían dos posibilidades: Utilizar una tarea asíncrona o utilizar un servicio.

Las tareas asíncronas se siguen ejecutando cuando se cambia la actividad (interfaz + controlador de interfaz) presente en la pantalla. Sin embargo, si la aplicación se cierra o a veces si se minimiza, las tareas asíncronas se pueden cancelar en mitad de su ejecución, por lo que podría ponerse en aprietos a los ficheros que se están mezclando o los resultantes de este proceso.

La otra opción disponible es la de los servicios. Aunque los servicios están pensados para correr en segundo plano de forma continua y las tareas asíncronas para procesos que acaben solos, los servicios tienen la ventaja de no morir aunque la aplicación se cierre. Únicamente la desinstalación de la aplicación o el sistema operativo pueden terminar el proceso.

Por otra parte, durante el desarrollo se encontró con un problema a la hora de transmitir los ficheros. En un principio se pensó en utilizar las funciones de transmisión de mensajes de texto entre los dispositivos, pero contaban con una limitación de unos 300 caracteres debido a la memoria destinada del reloj para esta tarea. Además, al enviar el contenido de los ficheros como texto, se perdía el formato, por lo que los ficheros del móvil resultaban ilegibles.

Como alternativa, se ha utilizado *ChannelApi*, un método no muy extendido en Android con el que poder transmitir ficheros entre dos dispositivos. Es una forma más pesada de

enviar datos, motivo por el cual Android no lo publicita mucho, pero se destina más memoria en el reloj para enviar estos ficheros, pudiendo llegar a transmitir ficheros de hasta aproximadamente 100KB. Por su parte, la aplicación cuenta con una curiosa limitación, los ficheros no mantienen su nombre ni su fecha de creación al ser recibidos en el móvil. Esto supone un gran problema, puesto que es mucho más pesado y lento leer todos los ficheros, obtener los *timestamps* y ordenar todos los datos que simplemente leer el fichero con la fecha más antigua, además de no saber de qué sensor se trata. Para solucionar este problema, se introdujo el nombre del sensor así como la fecha y hora de creación del fichero para leerlo en cuanto se reciba el fichero y renombrarlo. Así, a la hora de mezclar los ficheros, únicamente hay que ordenar por la hora de creación de los ficheros y escribir en el fichero que corresponda al sensor que esté indicado en el nombre.

4.4.4 - La comunicación reloj-móvil

La comunicación entre el reloj y el móvil es fundamental para transmitir los ficheros desde el reloj al móvil. El protocolo diseñado para la transmisión de los ficheros debe permitir la transmisión de los ficheros de forma secuencial desde un solo reloj al mismo tiempo hasta el móvil.

En este protocolo se han utilizado dos tecnologías diferentes: La de mensajes y la de envío de ficheros. En el caso del móvil, este se comunica con el reloj enviando mensajes vacíos, utilizando un *path* que acompaña a cada mensaje a modo de comando. Con estos comandos el móvil indica al reloj si debe enviarle el número de ficheros a transmitir o si debe enviar un fichero. Por parte del reloj, este utiliza mensajes para enviar el número de ficheros que tiene por enviar y *channelApi* para enviar los ficheros.

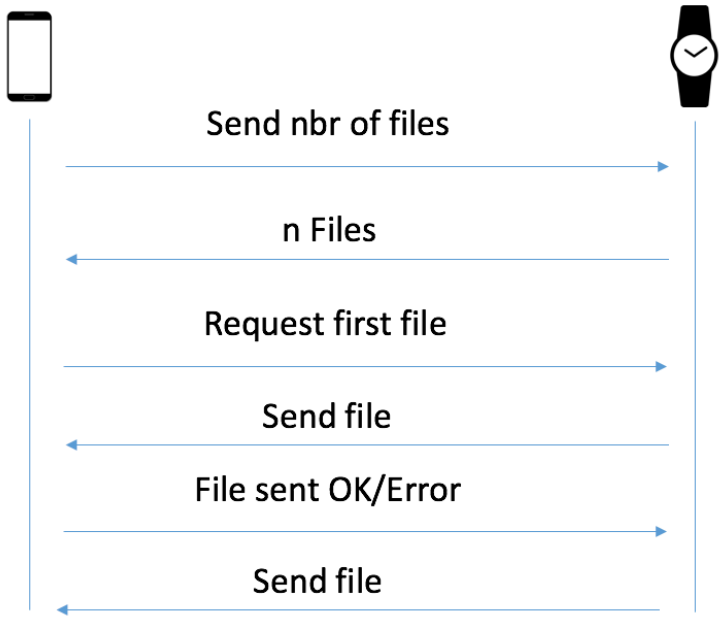

Ilustración 15: Protocolo de comunicación Wear - Mobile

Los pasos que siguen los dos dispositivos son los siguientes:

1. El móvil envía un mensaje pidiendo el número de ficheros que tiene el reloj para enviar.

2. El reloj responde con otro mensaje en el que indica los ficheros disponibles.

3. El móvil, en caso de que el número no sea 0, envía un mensaje pidiendo el primer fichero disponible.

4. El reloj envía por un canal de *ChannelApi* el primer fichero disponible.

5. Si el móvil ha recibido el fichero correctamente, envía un mensaje pidiendo que se envíe el siguiente fichero. Si ha habido algún error, el móvil envía otro mensaje distinto para que se envíe otro fichero.

6. Al recibir el mensaje de que el fichero ha sido enviado correctamente, el reloj lo marca como enviado y envía el fichero siguiente. Si ha habido un error, marcará el fichero como no enviado, por si se puede recuperar más adelante tal y como ha ocurrido, y se enviará el siguiente.

7. Los pasos 5 y 6 se repiten hasta que se han enviado todos los ficheros.

Lo primero que se puede apreciar en este protocolo es la ausencia de un mensaje final que indique que se ha terminado el envío de datos. Esto es debido a que no es necesario. El API de Android para recibir mensajes o ficheros hace que la aplicación esté siempre pendiente de si eso pasa, por lo que no es necesario cerrar la comunicación, al hacerse automáticamente. Por otra parte, la comunicación entre los dos dispositivos, al realizarse a través de Bluetooth, es muy pesada y consume muchos recursos, en especial para el reloj, así pues es conveniente evitar en todo lo posible estas comunicaciones.

Otro punto a tener en cuenta es el de la memoria RAM del dispositivo menos potente, el reloj. Aunque el API de Android está pensado para enviar ficheros o mensajes de cualquier tamaño, estos envíos están limitados por la memoria que el sistema operativo da a la aplicación para el envío, la cual es dependiente de la memoria libre en RAM. Esto ha implicado que los ficheros deban ser más pequeños a la hora de generarse en el reloj, y por eso se cierran y se crean nuevos ficheros de medidas cada pocos minutos.

Aunque se ha intentado limitar el tiempo en el que los fichero están abiertos para ser escritos, lo cierto es que a veces se recopilan muchos más datos de los esperados, engordando los ficheros. Esta es la principal razón de que algunos ficheros no se envíen, que la memoria de la cola de envío se llene y el fichero sea transmitido incompletamente. Otro problema que ha causado en más de una ocasión este acontecimiento es el cuelgue de la aplicación, puesto que al superarse la memoria de la cola y no poder enviarse el fichero bien, se bloquea el hilo de la aplicación que lo enviaba y el sistema operativo cierra la aplicación. Este problema es resuelto más adelante, en la aplicación de autorregulación, con un protocolo mejorado de comunicación.

4.4.5 - Pruebas

Aunque a medida que se ha ido desarrollando la aplicación de recolección de datos se han ido probando las funcionalidades, se han llevado a cabo una serie de pruebas para comprobar su correcto funcionamiento.

Una de las primeras pruebas que se hizo fue la de la identificación de los relojes. Al poder tener varios relojes emparejados de forma simultánea con el móvil se hace necesario identificar de forma única cada uno de los relojes. Además esta identificación tiene que ser visible de forma manual en el reloj mediante alguna opción que ofrezca el sistema operativo y obtenible programáticamente. En un principio, como se ha dicho anteriormente, se decidió probar con la MAC, pero tras realizar la prueba se pudo comprobar que sin conexión a internet, Android no la devuelve. Por ese motivo se volvió al identificador de la API de Android, que aunque no sea exclusivo tiene baja probabilidad de repetirse.

Otra de las pruebas que se hizo fue la de entregar un par de relojes al personal del Amilab para que probasen la aplicación durante un fin de semana entero. Uno de los relojes se apagó por acabarse la batería, lo cuál provocó que se descubriese lo que aquí se considera un importante fallo de los relojes de Android Wear: La pérdida de la hora. Al no disponer de un reloj interno, cuando los relojes son apagados o se acaba la batería pierden la hora y no la pueden recuperar por sí mismos, teniendo que conectarse al móvil con el que están emparejados para obtener su hora.

Una de las pruebas más importantes ha sido la que ha forzado la memoria de transmisión de datos. Como se ha descrito anteriormente, los ficheros que se pueden enviar y recibir está limitados por la memoria que el dispositivo tiene para transmitir ficheros. Tras comprobar en el laboratorio que algunos ficheros se transmitían parcialmente, se pasó a realizar una serie de pruebas en las que se enviaban de forma continua varios lotes de archivos cada vez más grandes. De esta forma se pudo determinar de forma aproximada cuál es el tamaño máximo de archivos que se puede enviar sin que haya posibilidades de corrupción, ya que ni el API de Android ni el fabricante permiten obtener esta cifra de otra forma.

Ya con la aplicación terminada, se ha pasado a la fase de pruebas en vivo, en la que se entregaron varios relojes al Instituto de Psico-Pediatría Dr. Quintero Lumbreras para que pusiesen relojes a varios alumnos. Las primeras semanas se visitó esta institución de forma frecuente para recoger los datos recopilados y comprobar que no existían fallos. Una vez considerada la aplicación suficientemente estable, se pasó a la fase de recolección de datos en el instituto mientras se comenzó el desarrollo de la segunda aplicación.

5 - Fase 2: Aplicación de regulación emocional

5.1 - Funcionalidad requerida

Esta aplicación para relojes Android Wear es, junto a la correspondiente versión para móvil Android, la versión final de TIC-TAC-TEA.

Su principal objetivo es mostrar una secuencia de imágenes, conocida como regulación, en la pantalla del reloj en el momento que se detecte una crisis en el usuario para que este pueda volver a su estado normal. Para la detección de las crisis se hace uso de las medidas biométricas devueltas por los sensores de los que disponen los relojes y de un perfil obtenido a través de las mediciones recopiladas con la aplicación de recogida de datos de la primera fase. Los cálculos para la detección de estrés se realizarán al finalizar una ventana de tiempo, de 30 segundos por defecto, que se repetirá hasta que el usuario detenga las mediciones.

Para refrescar conceptos, se recuerda que una regulación es una secuencia de estrategias, las cuales son una serie de imágenes estáticas o dinámicas mostradas de forma secuencial y una serie de reglas sobre cómo deben ser mostradas. Cada estrategia define cómo puede ser la navegación por las imágenes: manual (a través de toques en la pantalla) o automática (según el tiempo establecido en la estrategia). En caso de ser automáticas, se podrá tener una animación que indique el paso del tiempo: Un círculo o cuadrado que bordee la pantalla del reloj o una animación que "inunde" la pantalla en función del porcentaje de tiempo pasado. Las imágenes utilizadas en las estrategias vendrán dadas desde la versión móvil, y se recopilarán bien desde la página de ARASAAC o bien mediante fotos. También se permitirá que la estrategia muestre todas las imágenes que la componen en un carrusel durante un tiempo establecido y se podrá añadir una pregunta al final de cada una que pida al usuario que confirme si se siente mejor.

La regulación se construirá en el móvil y será personalizado en cada usuario. Por tanto, la aplicación del reloj debe ser capaz de interpretar la regulación trasferida desde el móvil y mostrarla tal cual al usuario con los recursos gráficos que también proporciona ese dispositivo.

De forma paralela, para seguir mejorando el perfil, la aplicación seguirá recopilando datos biométricos que transmitir al móvil. Esta funcionalidad sigue siendo requerida, aunque aún no esté definido si finalmente irá en la aplicación que se entregará al cliente.

Otra funcionalidad nueva es la de la recopilación de eventos. Estos eventos incluyen la detección del inicio y final de la regulación o la interacción del usuario con la aplicación, especialmente con la regulación. Una forma de medir la efectividad de la regulación es estudiando la interacción de los usuarios con las actividades propuestas en esta, como puede ser toques en la pantalla, los gestos para minimizar la aplicación o la respuesta escogida a las preguntas planeadas en la regulación. Por ese motivo, esta aplicación recopila esos eventos y los guarda en ficheros de texto que serán transferidos al móvil para su estudio.

5.2 – Revisión de Requisitos

En este apartado se va desarrollar con más detalles cuáles son los requisitos de la aplicación de regulación. Aunque esta aplicación ha empezado a desarrollarse mucho más tarde, cuando la aplicación de recopilación estaba terminada y ya se tenían las primeras medidas, la aplicación siguió su desarrollo en un entorno de constante investigación, puesto que la documentación seguía siendo limitada y las características de los relojes no han sido del todo descubiertas. Como se verá más adelante, algunas funcionalidades existentes en la aplicación de recopilación han sido mejoradas precisamente por descubrir y probar formas mejores de realizarlas.

Por otra parte, mientras se estaba desarrollando esta aplicación, la de recopilación seguía siendo probada en el Instituto de Psicopediatría, por lo que con todo el feedback recibido se ha modificado algunos requisitos iniciales. También se ha de tener en cuenta que algunas de las peticiones iniciales se han cambiado por otras a petición del personal del IPP por ajustarse más a lo requerido. En este documento se refleja la versión final de los requisitos que tendrá la aplicación, algunos de los cuales son similares a los de la primera aplicación.

5.2.1 - Requisitos funcionales

Para empezar, este sistema contará con un módulo dedicado a la recolección de datos a partir de los sensores que dispone el reloj:

- RS1: El reloj deberá recopilar los datos proporcionados por el sensor cardiaco. Esos datos serán la media de pulsaciones del último periodo de tiempo así como la precisión de la medida tomada. Esta medida irá desde 0 (No se están tomando las pulsaciones bien porque no hay contacto con el cuerpo del usuario) hasta 3 (las medidas parecen estar realizándose con total precisión).
- RS2: El reloj deberá recopilar los datos proporcionados por el detector de pasos. La medida se tomará cada vez que se realice un movimiento interpretado como un paso. Esta medida irá acompañada de su precisión, siendo 0 (No se están tomando medidas), 1 (El movimiento apenas se asemeja a un paso), 2 (El movimiento se asemeja a un paso pero no es seguro) o 3 (El movimiento probablemente sea el propio de un paso).
- RS3: El reloj deberá recopilar los datos proporcionados por el sensor de movimiento, acelerómetro, el cuál devolverá la cantidad de movimiento detectada en coordenadas X, Y, y Z cada cierto tiempo. La medida deberá ir acompañada de su precisión, empezando en 0 (la medida es errónea) y terminando en 3 (la medida está tomada a máxima precisión).
- RS4: El reloj deberá recopilar los datos proporcionados por el giroscopio. Esta medida se tomará cada cierto periodo en coordenadas X, Y y Z que expresen cuanto ha girado el dispositivo respecto a la posición anterior. La medida irá acompañada de su precisión, desde 0 (La menor precisión) hasta 3 (máxima precisión de la medida).
- RS5: Los sensores estarán configurados para que devuelvan las medidas cada cierto periodo, en función de cada sensor y de la configuración del fabricante con el sistema operativo. En Android existen los siguientes modos de recopilar las medidas tomadas por los sensores:

    - Modo de un solo disparo: Este modo devolverá la primera medida tomada. Obviamente esto no sirve si se quiere realizar un seguimiento continuo del usuario del dispositivo.
    - Modo continuo: Las medidas son devueltas constantemente cada cierto periodo en función de la configuración del sensor definida por el fabricante. El problema de este modo es que puede devolver muchos valores nulos por no tener nuevas mediciones en el último periodo. Por ejemplo, si no existen pasos en los últimos 500 ms, el sensor de pasos devolverá 0 pasos. Tras dilucidarlo, se ha decidido que este tipo de medidas son inválidas, por tener medidas poco útiles o por tener un periodo demasiado grande para algunos sensores, como el giroscopio.
    - Modo de detección de cambios: Este modo devolverá medidas cada vez que el sensor registre un cambio en las medidas tomadas. Se ha rechazado este modo puesto que en sensores como el cardíaco o el detector de pasos esto supone que si no hay variación en sus respectivas medidas (pulso o pasos por unidad de tiempo) no se notificará el estado actual de dichas medidas. Por otra parte, en sensores con mucha precisión como el acelerómetro o el giroscopio se asume un cierto margen de variación para considerar que ha habido cambio en las medidas, lo cual supone un problema en caso de movimiento o giros constantes, en donde no hay suficiente variación como para notificar el cambio.
    - Modo disparador especial: Este modo combina los mejor los dos modos anteriores. Por un lado, en sensores de gran precisión como el giroscopio, el acelerómetro o el pulsímetro devuelve de continuamente mediciones aunque las medidas sean iguales o similares. Por otro lado, en sensores como el detector de pasos solo devuelve la detección de pasos cuando estos se producen. Se ha considerado que este es el modo óptimo para la actividad que se va a realizar.
- RS6: Dada la variedad de dispositivos Wear, la aplicación debe estar preparada para seguir funcionando en caso de que alguno de los sensores no esté disponible en el dispositivo. Esto tiene un claro efecto negativo a la hora de dilucidar si se está teniendo una crisis o no, puesto que si falta un sensor clave como el cardíaco, podrían darse falsos positivos o falsos negativos.
- RS7: La aplicación debe ser capaz de evitar que los sensores se apaguen al ponerse la aplicación en segundo plano o cerrarse. Android Wear deja en segundo plano la aplicación al pulsar el botón inicio, al desplazar el dedo de izquierda a derecha o al tapar la pantalla. Debido a la poca potencia de estos dispositivos, al realizar cualquier acción con el reloj, la aplicación se cierra. La aplicación debe ser capaz de que los sensores sigan recopilando datos en cualquiera de estas situaciones.
- RS8: La aplicación dejará de recopilar datos biométricos de forma automática al poner los relojes a cargar o cuando la batería se encuentre en un nivel muy bajo. La razón para hacer esto no es otra que evitar la pérdida de los datos por corrupción de ficheros al cerrarse estos de forma abrupta porque se cierre la aplicación. Para cumplir este objetivo se hará uso de una escucha del evento de batería.
- RS9: La aplicación debe evaluar con los datos recogidos en los últimos 30 segundos si el usuario ha entrado en un episodio estrés o no. A este periodo de tiempo, que puede variar según los resultados obtenidos en las pruebas, se le conoce como ventana de tiempo.

- RS10: En caso de que el usuario tenga una crisis, la aplicación deberá iniciar la regulación establecida en el reloj.
- RS11: Justo antes de iniciar la regulación, el reloj debe vibrar para llamar la atención al usuario. En caso de que no se disponga de esta característica se deberá iniciar la regulación sin más.

Una de las funcionalidades totalmente nuevas de esta aplicación es la de la regulación, que cuenta con un módulo propio:

- RR1: La aplicación debe ser capaz de interpretar el fichero JSON de configuración de la regulación y cargar en memoria los pasos a seguir si el fichero es correcto.
- RR2: La aplicación debe ser capaz de reproducir la secuencia de imágenes o GIFs que forman cada una de las estrategias, localizando los recursos descritos en el fichero de configuración de la regulación.
- RR3: La aplicación debe ser capaz de reproducir las estrategias que conforman la regulación en el orden especificado por el fichero de configuración.
- RR4: La aplicación debe ser capaz de mostrar imágenes en formato JPG y PNG.
- RR5: La aplicación debe ser capaz de mostrar imágenes dinámicas en formato GIF.
- RR6: La aplicación debe ser capaz de detectar toques en la pantalla y realizar la transición de una imagen a otra solo si la estrategia está configurada para que la transición entre sus imágenes sea manual. En caso contrario debe ignorar cualquier toque en la pantalla.
- RR7: La aplicación debe cambiar de una imagen a otra tras los segundos especificados por la configuración si la estrategia indica que la transición entre los pasos (imágenes) de la estrategia debe hacerse por tiempo.
- RR8: La aplicación debe mostrar una barra de progreso que bordee la pantalla de reloj, sea circular o cuadrada, cuando la configuración de la animación del tiempo indique que se trata de una animación tipo “donut”. El porcentaje de la barra de progreso será proporcional al tiempo pasado tras empezar la estrategia.
- RR9: La aplicación debe mostrar el paso del tiempo en estrategias con transición automática mediante una animación de “inundación” cuando la configuración de la estrategia indique que es este tipo de animación. El nivel de “inundación” de la pantalla será proporcional al tiempo pasado desde que se inició la estrategia, estando la pantalla vacía cuando el progreso es 0 y llena cuando hayan pasado todos los segundos de la estrategia.
- RR10: La aplicación debe hacer vibrar el reloj cuando se detecte estrés.
- RR11: La aplicación debe restaurar la regulación en el punto en el que se había quedado cuando el usuario salga de la aplicación. La regulación solo se dará por terminada cuando se llegue al final de la última estrategia.
- RR12: Además de la secuencia de estrategias, la regulación podrá contar con conjuntos de estrategias. Estos conjuntos se mostrarán como un selector para elegir la siguiente estrategia a realizar.
- RR13: Las estrategias que así estén configuradas podrán mostrar todas sus imágenes como un carrusel en lugar de realizar una secuencia.
- RR14: La aplicación debe asegurarse de que las estrategias configuradas como un carrusel sean estrategias de tiempo, con una duración en pantalla limitada.

- RR15: La aplicación deberá ser capaz de mostrar por pantalla preguntas simples para pedir al usuario que exprese su estado. Las preguntas serán de respuesta SI/NO y mostrarán un botón con un tic verde y una cruz roja para las opciones.
- RR16: Si la estrategia tiene una pregunta y el usuario escoge la opción considerada como correcta por el tutor, se mostrará una imagen de refuerzo positivo si la estrategia está configurada de esa manera.
- RR17: La aplicación debe detener la regulación si el reloj es puesto a cargar.

Otra funcionalidad nueva añadida a la aplicación de autorregulación es la que se desarrolla en parte en el módulo de eventos. Este módulo se encarga de recoger eventos como las interacciones del usuario con la aplicación.

- RE1: La aplicación debe recoger el inicio de una regulación debido a la detección de estrés.
- RE2: La aplicación debe recoger la finalización de la regulación.
- RE3: La aplicación debe recoger la valoración del estado del usuario en función de sus datos biométricos. Es decir, se guardará el resultado de la decisión acerca de si el usuario tenía o no estrés.
- RE4: La aplicación deberá detectar toques en pantalla cuando la aplicación esté activa. Estos toques incluirán aquellos que hacen avanzar las estrategias cuando sus transiciones son manuales y los toques que no tienen efecto debido a que las transiciones son automáticas por tiempo.
- RE5: La aplicación detectará qué respuesta se ha elegido ante las preguntas de SI/NO establecidas en la regulación al tocar el botón correspondiente a la respuesta.
  *En caso de tocar fuera del área de los botones se contará como el toque establecido en el RE4.
- RE6: Los gestos de cierre con *swipe*, desplazar el dedo de izquierda a derecha para cerrar la aplicación, deberán ser detectados y recogidos.
- RE7: Los eventos que suspendan la aplicación que no sean el *swipe*, como pulsar el botón de inicio o tapar la pantalla con la mano, deberán ser recogidos también.
- RE8: La aplicación detectará siempre el cierre o suspensión de la aplicación, pero además deberá detectar cuando se ha salido de la regulación sin haberse terminado.
- RE9: Se registrará un nuevo evento cuando la regulación llegue a su fin natural.
- RE10: Cuando se inicie una estrategia o uno de los pasos, imágenes, que lo componente, se registrará como un evento.
- RE11: Los movimientos de *swipe* de los carruseles se clasificarán como un evento aparte, separado de los otros casos de *swipe*.
- RE12: La aplicación deberá detectar y recopilar el *scroll* realizado en los selectores de estrategias.
- RE13: Al mostrarse una imagen de refuerzo positivo se registrará un evento indicando su aparición.

Por otra parte, el módulo de almacenamiento ha cambiado ligeramente sus funcionalidades. Ahora, además de guardar los datos de las mediciones, también guarda los eventos de la aplicación:

- RA1: Cada medida o evento que se vaya a almacenar debe estar acompañada de un timestamp.
- RA2: El almacenamiento de las medidas y los eventos se realizarán en ficheros de texto plano.
- RA3: Cada uno de los sensores contará con un fichero de texto diferente. Los eventos estarán todos en un mismo fichero de texto.
- RA4: Los ficheros deberán estar estructurados para su fácil visualización y procesado. Los ficheros se organizarán en columnas separadas por un tabulador. En el caso de los ficheros de medidas cada una de esas columnas será el *timestamp*, la precisión y el/los componentes de cada medida. Para los ficheros de eventos, será un *timestamp*, el evento y una tercera columna para aquellos eventos que necesiten más información, como el nombre de la estrategia que se va a empezar.
- RA5: Los ficheros de los sensores contarán con una filas a modo de cabecera de cada archivo. En la primera se detallarán los datos del sensor y el nombre del fichero.
- RA6: Dada la facilidad con que el sistema puede cerrar la aplicación, para evitar ficheros muy pesados que carguen mucho la memoria RAM del dispositivo y para disminuir las pérdidas en caso de ficheros corruptos, los ficheros de las medidas y de los eventos se irán troceando en ficheros pequeños que se juntarán más tarde fuera de este dispositivo.
- RA7: Los ficheros se borrarán una vez se hayan transmitido a la aplicación móvil y no solo cuando se desinstale la aplicación, como pasaba en la aplicación de recolección.
- RA8: El nombre del fichero deberá contener el id del reloj, el nombre del sensor o la palabra “eventos” y la fecha y hora de creación del fichero.

Al no existir ninguna forma manual de acceder a los archivos guardados en el reloj, debe existir un módulo que sea capaz de transferir los ficheros de las mediciones y los eventos al móvil, donde se pueden extraer con un explorador de archivos. Este módulo debe ser también capaz de recibir el fichero de configuración de la regulación y los recursos que se utilizarán durante la misma, sean imágenes o GIFs:

- RT1: La aplicación debe ser capaz de transmitir los ficheros guardados en el reloj sin enviarlos por duplicado.
- RT2: En caso de error al enviar un fichero, se debe reintentar enviarlo. En caso de no poder en el segundo intento, se deberá continuar con el siguiente fichero.
- RT3: La aplicación debe ser capaz de recibir el fichero de configuración de la regulación.
- RT4: La aplicación debe ser capaz de recibir los ficheros de los recursos. En caso de que estos ficheros vengan troceados, debe ser capaz de mezclarlos y formar el fichero.

Por otra parte, como se comentó anteriormente, la aplicación está dividida en reloj (El principal objetivo de este TFM) y en móvil (objetivo de otro TFM). Sin embargo, para asegurarse del buen funcionamiento del protocolo de transmisión de ficheros y su almacenamiento en el móvil, se creará una aplicación móvil de prueba que contenga dichas funcionalidades. La aplicación móvil contará con dos módulos.

Un módulo de transmisión de ficheros con los siguientes requisitos:

- RMT1: La aplicación debe ser capaz de notificar al reloj cuándo puede enviar o recibir los ficheros.
- RMT2: La aplicación tiene que indicarle al reloj si se ha recibido correctamente el último fichero o si ha habido algún error.
- RMT3: Para evitar que la transmisión de ficheros se haga muy larga, se debe mostrar una barra de progreso que indique cuántos de estos se han recibido.
- RMT4: La transmisión de ficheros entre móvil y reloj se hará necesariamente por Bluetooth, puesto que no todos los relojes poseen Wifi y el centro en el que se realizarán las pruebas no posee dicha instalación.
- RMT5: La aplicación debe ser capaz de enviar el fichero de configuración de la regulación y los recursos asociados a esta.
- RMT6: La aplicación debe ser capaz de recibir los ficheros de medidas y de eventos.

* Queda fuera de los requisitos de la aplicación la funcionalidad para que el móvil solo se comunique con el reloj seleccionado y con otros con los que se hubiese emparejado. Esto se debe a que la gestión de emparejamiento, conexión y selección del reloj exclusivamente se puede hacer desde la aplicación Android Wear.

También se incluye un módulo encargado del almacenamiento de ficheros:

- RF1: La aplicación tiene que ser capaz de almacenar los ficheros en una carpeta accesible desde un explorador de archivo para móviles.
- RF2: La aplicación debe ser capaz de recoger los ficheros recibidos y mezclarlos quedando repartidos en una carpeta para cada reloj y un fichero para cada tipo de medidas o los eventos. Para su ordenación, se hará uso de la fecha y hora contenidos en el nombre del fichero.
- RF3: Los ficheros recibidos parciales deben ser borrados una vez se haya terminado la mezcla.
- RF4: Al realizar la mezcla de ficheros, la aplicación debe hacerlo en orden cronológico de las medidas o eventos.

Por último, la aplicación cuenta con un módulo para la realización de cálculos sobre las medidas tomadas que necesitará el perfil generado para averiguar si el usuario está en crisis o no:

- RC1: La aplicación debe ser capaz de calcular la media de las medidas o la componente de las medidas que se le especifique.
- RC2: La aplicación debe ser capaz de calcular la mediana de las medidas o la componente de las medidas que se le especifique.
- RC3: La aplicación debe ser capaz de calcular la varianza de las medidas o la componente de las medidas que se le especifique.
- RC4: La aplicación debe ser capaz de calcular la desviación estándar de las medidas o la componente de las medidas que se le especifique.
- RC5: La aplicación debe ser capaz de calcular el rango intercuartílico de las medidas o la componente de las medidas que se le especifique.

- RC6: La aplicación debe ser capaz de calcular el valor cuadrático medio de las medidas o la componente de las medidas que se le especifique.
- RC7: La aplicación debe ser capaz de calcular la desviación media absoluta de las medidas o la componente de las medidas que se le especifique.
- RC8: La aplicación debe ser capaz de calcular la regresión polinómica de las medidas o la componente de las medidas que se le especifique.
- RC9: La aplicación debe ser capaz de calcular la pendiente de la función dibujada por las medidas o la componente de las medidas que se le especifique.
- RC10: La aplicación debe ser capaz de calcular la tendencia de las medidas o la componente de las medidas que se le especifique.
- RC11: La aplicación debe ser capaz de calcular la magnitud de cambio de las medidas o la componente de las medidas que se le especifique.

5.2.2 - Requisitos no funcionales

Por otra parte, se ha puesto como objetivo los siguientes requisitos no funcionales, comunes a ambas aplicaciones:

- RNF1: Interfaz simple. La aplicación debe tener una interfaz fácil de utilizar. En el reloj se debe mostrar únicamente el botón de start-stop para iniciar o detener la mediciones y la regulación asignada. En el caso del móvil, se mostrará el botón para iniciar la recogida de datos y el botón de enviar una regulación de prueba.
- RNF2: Cumpliendo con la anteriormente mencionada ley de protección de datos, los datos biométricos permanecerán siempre en los dispositivos que utilicen aplicaciones de TIC-TAC-TEA y no saldrán al exterior de ninguna forma.
- RNF3: Los datos no deben poder relacionarse con ninguna persona en concreto (anonimización).
- RNF4: Previsión ante acciones no intencionadas. Debido a que la aplicación del reloj será utilizada por niños, la aplicación debe seguir funcionando con normalidad pese a agitaciones del reloj, tapar la pantalla o desplazar el dedo de izquierda a derecha. Estas acciones cierran las aplicaciones o apagan la pantalla. También es importante que la aplicación deje de recopilar datos o detenga la transmisión de ficheros en el momento en el que se detecte que el reloj está cargándose o con poca batería. Esto último es aplicable también para el móvil.
- RNF5: Diseño *responsive*. La aplicación *mobile* debe ser usable tanto en móviles como en tabletas. La aplicación Wear debe funcionar en relojes cuadrados y circulares.

5.3 - Diseño de la aplicación

5.3.1 - Diseño del sistema

Al igual que la primera aplicación, la aplicación TIC-TAC-TEA de autorregulación está dividida en dos partes: Una aplicación para *smartwatches* Android y otra para móviles con el mismo sistema operativo.

La aplicación para relojes es el objetivo principal de este TFM. Entre sus funcionalidades se incluyen la recopilación de datos biométricos con los que realizar un perfil que detecte crisis y, una vez perfilado, detectar cuando se producen esas crisis. Además de recopilar datos biométricos, la aplicación recoge datos como las interacciones del usuario con la aplicación, con el objetivo de averiguar cómo se usa la aplicación y mejorar su adecuación a cada persona. Una vez se detecte una crisis, la aplicación inicia una secuencia de imágenes y GIFs llamada regulación. Esta secuencia viene data por un archivo de configuración recibido desde la aplicación móvil, al igual que lo recursos que se utilizan en dicha secuencia.

La aplicación móvil es la encargada de recoger los archivos de reloj y enviar los correspondientes de la regulación a ese dispositivo. Como el diseño y desarrollo de esta aplicación es parte de otro TFM, en este trabajo solo se ha desarrollado el sistema de transmisión de ficheros, que difiere del de la aplicación de recopilación.

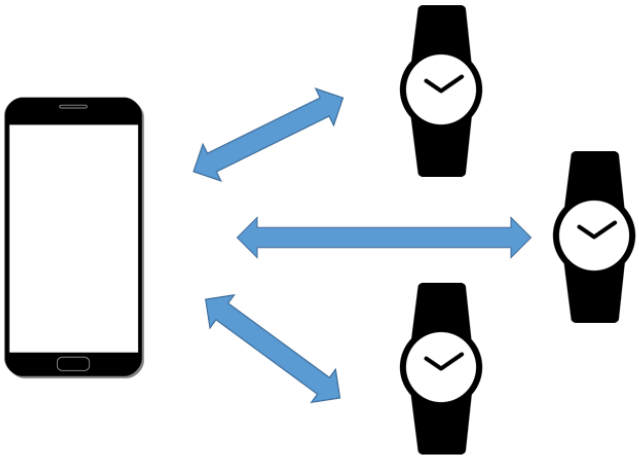

Ilustración 16: Conexión móvil - relojes

Como se ve en el diagrama anterior, el móvil vuelve a tener una funcionalidad similar a la de un servidor, por recoger y alojar los archivos de medidas, logs y todos los recursos utilizados durante la regulación. Todos esos ficheros deben ser accesibles mediante un explorador de archivos, por lo que se utilizarán carpetas externas para poder recoger esos archivos y analizarlos.

Aplicación Android Wear

La aplicación para relojes es la encargada de recopilar y almacenar datos biométricos y los eventos que incluyen la interacción del usuario y de enviarlos cuando el móvil con el que está emparejado el reloj lo requiera. También es la encargada de mostrar la secuencia de autorregulación. Estas funcionalidades son las que definen los subsistemas que conformarán la aplicación de TIC-TAC-TEA para reloj.

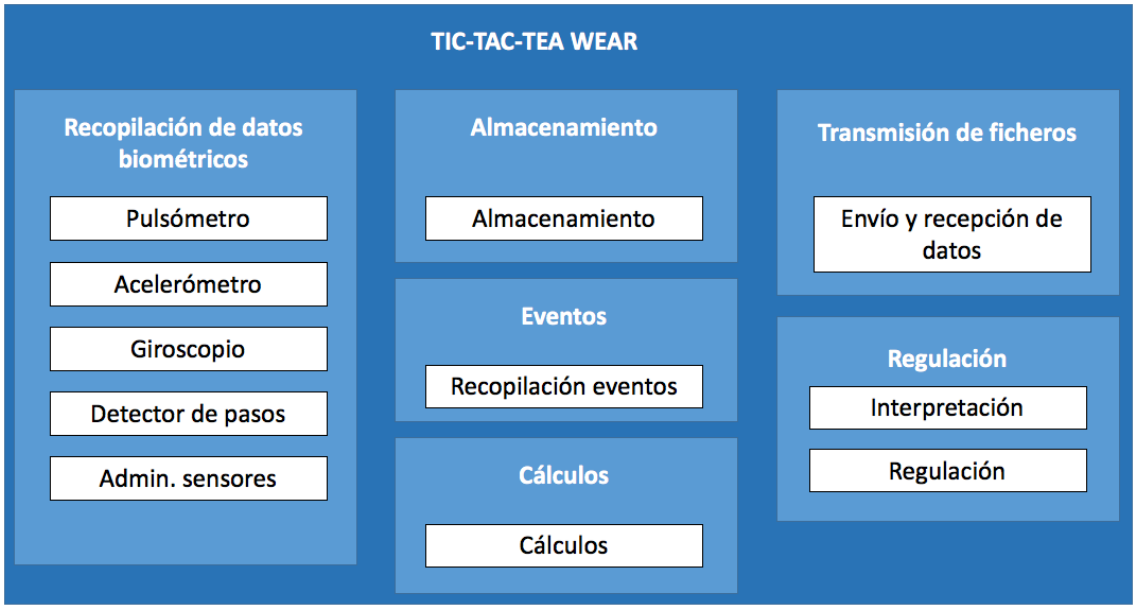

Ilustración 17: Módulos de TIC-TAC-TEA Wear

- Subsistema de recopilación de datos biométricos: Es el encargado de manejar cada uno de los sensores del dispositivo y la recolección de los datos que estos arrojan.

    - Módulo del sensor cardíaco. Es el encargado de recopilar los datos del sensor cardíaco. Este sensor medirá cada cierto tiempo las pulsaciones detectadas desde la medición anterior, o desde el encendido del sensor si se trata de la primera medición, y establecerá la media por minuto en cada medición. Por ejemplo, en caso de detectar en un periodo de 5 segundos 10 pulsaciones, la media de pulsaciones por minuto sería de 120 pulsaciones/minuto. Las medidas por minuto son las que se devuelven a la aplicación. Por otra parte, las medidas irán acompañadas de un *timestamp* y un número que indica la precisión de la medida tomada (siendo 0 la menor precisión y 3 la máxima precisión).

    - Módulo del sensor de pasos. Este módulo es el encargado de recoger los movimientos detectados como un paso cada vez que se produzcan. A diferencia del sensor anterior, en este se recoge el *timestamp* y la precisión únicamente cuando se produce este evento. En este caso la precisión de 0 a 3 corresponde a la similitud del movimiento del dispositivo al balanceo de los brazos en un paso.

    - Módulo del acelerómetro. Es el módulo que gestiona la recolección de los datos arrojados por el acelerómetro. Este sensor devuelve mediciones cada

cierto tiempo cuando existe una variación suficiente de movimiento. La configuración de la frecuencia con la que devuelve medidas y la sensibilidad mínima para considerarse movimiento viene dado por el fabricante del sensor y el fabricante del reloj. Las medidas vendrán en sus componentes X, Y y Z y estarán acompañadas de la precisión de la medida realizada y del tiempo. La precisión se moverá en el intervalo de números enteros desde 0, la menor precisión y posiblemente una medida errónea, a 3, la máxima precisión.

- Módulo del giroscopio. Es el módulo que gestiona la recolección de los datos devueltos por el sensor del giroscopio. Este sensor devuelve mediciones cada cierto tiempo cuando existe una variación suficiente de movimiento giratorio tomando como punto de referencia a la hora de medir giros el centro del sensor ubicado en el reloj. La configuración de la frecuencia con la que devuelve medidas y la sensibilidad mínima para considerarse movimiento viene dado una vez más por el fabricante del sensor y el del reloj. Las medidas vendrán en sus componentes X, Y y Z y vendrán siempre con una medida de la precisión de la medida realizada (de 0 a 3, siendo 0 la menor precisión y 3 el máximo) y del tiempo.

- Módulo de gestión de sensores. Es un módulo que se encarga de gestionar el ciclo de vida de los sensores, además de obtener sus medidas y utilizarlas en los cálculos del perfil para detectar estrés. El momento de utilizar las medidas de los sensores viene dado por la finalización de la ventana de tiempo, administrada desde este mismo módulo. Cuando la ventana de tiempo ha finalizado, se reiniciará y se volcarán las medidas tomadas en cuatro ficheros de texto, uno por cada sensor.

- Subsistema de eventos. Es el subsistema que contiene las clases que recopilan los distintos eventos de la aplicación, como son las interacciones del usuario o el estado actual de la regulación. Parte de la funcionalidad de este subsistema se encuentra implementado en las clases que forman el subsistema de regulación, puesto que la interacción con el usuario solo puede ser recogida por la clase correspondiente a la actividad que se muestra por pantalla.

  - Módulo de recolección de eventos. Es el que recibe los eventos que se están recopilando, los clasifica según el tipo de evento y prepara el texto que se guardará. Los diferentes tipos de eventos, como tocar la pantalla, hacer *swipe* o iniciar una estrategia, están definidos en este módulo.

- Subsistema de almacenamiento. Es el encargado de guardar las medidas de los sensores y los eventos en ficheros de texto. Cuenta con un solo módulo:

  - Módulo de almacenamiento. Guarda las medidas de los sensores y los eventos en ficheros de texto. Cada sensor contará con sus propios ficheros, mientras que todos los eventos se guardarán en un fichero de eventos común. Para evitar la posible pérdida de datos en caso de que alguno de los ficheros se corrompa porque el sistema se cargue demasiado y termine con procesos en segundo plano como el que recolecta medidas, los ficheros se seguirán troceando y no se guardarán en un solo fichero. Esto

es más probable sabiendo que la aplicación debe estar corriendo minimizada, lo que significa que no es una prioridad para el sistema operativo.

- Subsistema de regulación. Es el módulo principal de esta aplicación, y el que más peso tendrá. Es el que interpreta el fichero de configuración y muestra la regulación tal y como está descrita:

    1. Módulo de interpretación. Es el fragmento de la aplicación que lee el fichero de configuración JSON de la regulación y guarda los datos en memoria.

    2. Módulo de regulación. Es el encargado de mostrar la secuencia de imágenes que componen las estrategias de la regulación y el encargado de mantener el estado actual de esta, incluyendo la funcionalidad para restaurar la aplicación en el punto en el que se encontraba si el usuario cierra la aplicación por accidente.

- Subsistema de transmisión. Este es el subsistema encargado de transferir archivos entre el reloj y el móvil al que está sincronizado.

    1. Módulo de envío y recepción de datos. Es el que, al recibir la orden desde el móvil, envía todos los ficheros de medidas y eventos creados hasta el momento al móvil. Por cada fichero enviado se recibirá una confirmación desde el móvil para pasar al siguiente archivo. En caso de que algún fichero no se pueda enviar, se reintentará una vez más. Una vez han sido enviados todos los ficheros, este módulo indicará al sistema de almacenamiento que borre los ficheros para liberar memoria en el reloj.

        Por otra parte, este módulo, tras recibir la correspondiente orden del móvil, borra los ficheros de la regulación existentes en el reloj y recibe el nuevo fichero de configuración de la regulación y todos los recursos que esta utilizará. En caso de que estos ficheros vengan troceados, este sistema los mezclará una vez recibidos todos los fragmentos.

- Subsistema de cálculos. Es el subsistema que realiza los cálculos.

    1. Módulo de cálculos: Es el que incluye las clases que contienen el código necesario para realizar los cálculos en un hilo en segundo plano.

Aplicación Android Mobile

Sabiendo que la aplicación de Android Wear es el principal objetivo a desarrollar en este TFM, la aplicación para Android Mobile tendrá como únicas funcionalidades las pruebas para el protocolo de transmisión de ficheros, el guardado de los ficheros de medidas biométricas y eventos del reloj y el enviado de las regulación. El diseño de la regulación y la generación del fichero de configuración correspondiente a esta forman parte de otro TFM desarrollado en paralelo, por lo que para realizar las pruebas se dispondrán de una serie de ficheros de configuración generados a mano y los recursos que se necesitan.

Al igual que en la aplicación de recopilación de datos, los ficheros de medidas y eventos que se reciban serán mezclados por el orden cronológico de sus entradas en un fichero por cada sensor o uno para todos los eventos. Cada grupo de ficheros estará separado en carpetas exclusivas para cada reloj que maneje la aplicación.

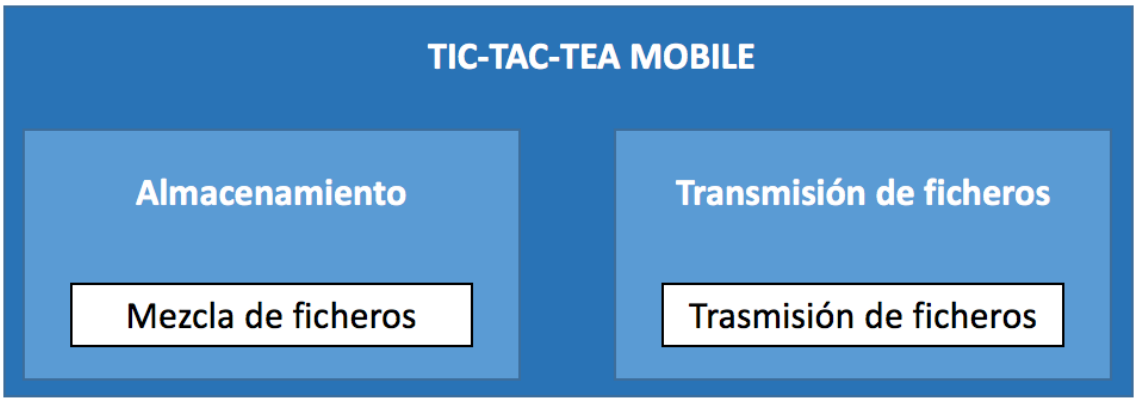

Ilustración 18: Módulos de TIC-TAC-TEA mobile

- Subsistema de transmisión de ficheros. Este subsistema es el que tiene como objetivo recibir todos los ficheros que tiene el reloj con el que está sincronizado y enviar los correspondientes de la regulación al dispositivo.

    1. Módulo de transmisión de ficheros. Una vez el usuario indique a la aplicación del móvil, a través de un botón situado en la pantalla, que se quiere recibir los ficheros, el móvil indicará al reloj sincronizado que debe enviar todos los ficheros que no haya enviado con anterioridad. Los archivos recibidos se guardarán directamente desde este módulo, puesto que lo que se recibe es el fichero en sí.

- Subsistema de almacenamiento. Es el encargado de almacenar los ficheros recibidos del reloj. Cuenta con el siguiente módulo:

    1. Módulo de mezcla de ficheros. Puesto que los ficheros se guardarán directamente desde el subsistema anterior, este subsistema solo contará con este módulo que se encargará de mezclar los ficheros por sensor o todos los eventos en orden cronológico y separando los ficheros de los distintos relojes.

Comunicación Mobile-Wear

Las aplicaciones que quieran comunicar datos entre un móvil o Tablet y un reloj son una misma aplicación que se divide en dos partes. La aplicación de TIC-TAC-TEA no es una excepción, también está dividida en dos partes: La aplicación para móvil y la aplicación para reloj. Debido a las limitaciones del sistema operativo del reloj, por razones de seguridad y manejo, Android Wear no permite que los archivos internos de las aplicaciones sean accesibles por otras apps, por lo que deben ser enviados al móvil desde la propia aplicación. La transmisión de los archivos, al igual que cualquier otro tipo de datos, se hará a través del Bluetooth, puesto que es una característica obligada de todos los relojes con Android Wear y todos los móviles capaces de comunicarse con relojes.

En esta ocasión se ha cambiado la forma se comunicar la aplicación del reloj y el móvil. Por una parte, los archivos que sean muy grandes para la memoria del reloj, como las imágenes, no se quedarán sin enviar, se trocearán en el dispositivo emisor y se juntarán sus partes en el receptor.

Por otra parte, ahora hay dos protocolos de comunicación. Uno para los ficheros de medidas y eventos (que envía ficheros desde el reloj al móvil) y otro para los ficheros de la regulación (que los envía desde el móvil al reloj).

Empezando con la comunicación de los ficheros de medidas y eventos:

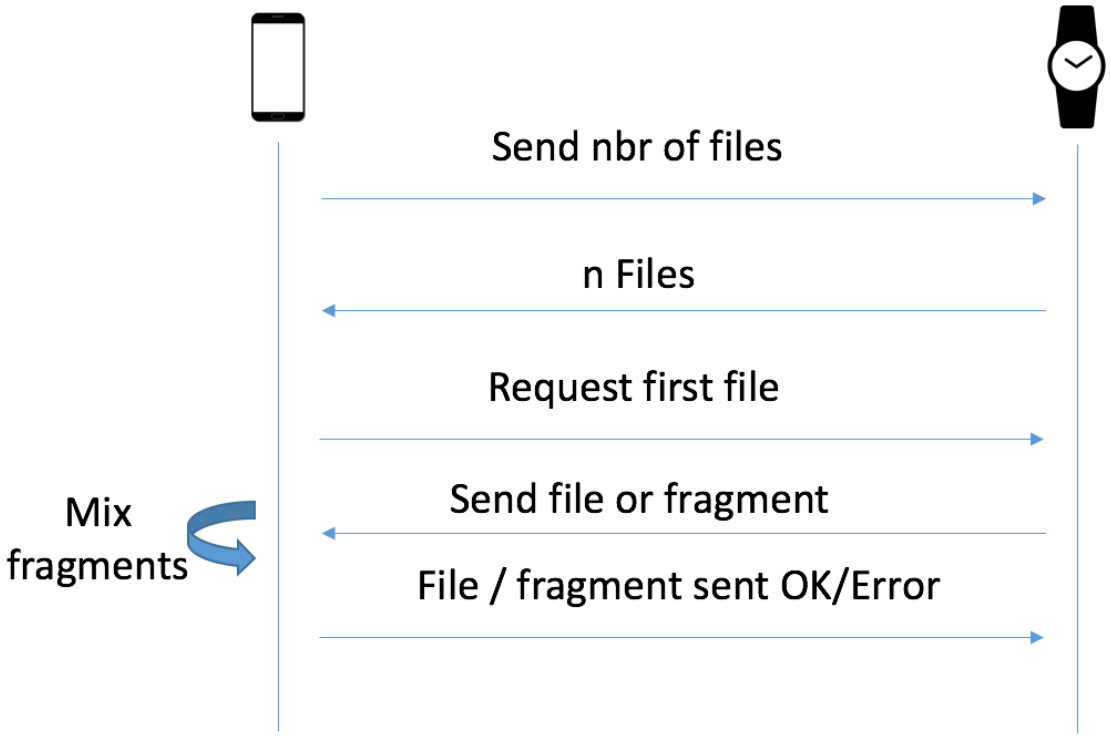

Ilustración 19: Envío de datos al móvil

Los pasos que deberá seguir este protocolo son los siguientes:

1. El móvil pregunta al reloj cuántos ficheros tiene para transferir.
2. El reloj responde con el número pedido. Este número se utilizará en el móvil para indicar al usuario de forma visual el progreso de la transmisión de los datos.
3. El móvil indica al reloj que inicie la transmisión de los ficheros que tenga.
4. El reloj envía el primer archivo no enviado. Si el fichero supera el umbral de los 100KB, se enviarán los primeros 100KB.
5. El móvil recibe el fragmento de fichero. Si es el último fragmento de fichero se mezclarán y se pedirá el siguiente archivo, si es un fichero entero también se pedirá el siguiente archivo y si es un fragmento pero no es el último del archivo, se pedirá el siguiente. En caso de error se pedirá que se reintente enviar.
6. El reloj envía el siguiente archivo si terminó con el anterior, el primer fragmento de 100KB si este fichero supera este tamaño. Si el fichero no ha terminado de enviarse se envía la siguiente tanda de hasta 100KB. Si ha habido un error se

reenviará el último fichero/fragmento. Si el error se repite con el mismo fichero/fragmento se considerará que el fichero está corrupto.

7. Los pasos 5 y 6 se repetirán hasta que se termine la transferencia de los datos. Al terminar, el reloj enviará un mensaje al móvil indicando que no hay más ficheros para que este muestre un mensaje de finalización.

Si la conexión Bluetooth se cierra la propia API de Google cerrará la conexión y se podrá reintentar la transmisión de ficheros después del último fichero enviado correctamente.

Para el envío de los ficheros de la regulación se utilizará un protocolo ligeramente distinto:

1. El móvil le pregunta al reloj si está libre, es decir, si no está ejecutando la regulación ni realizando mediciones.
2. Si el reloj está libre, borrará la regulación que tiene almacenada y enviará un mensaje afirmativo.
3. El móvil envía al reloj cuántos ficheros tiene para transferir y el fichero de configuración. Este número se utilizará en el móvil para indicar al usuario de forma visual el progreso de la transmisión de los datos.
4. El reloj pide el primer fichero.
5. El móvil envía el primer fichero disponible. En caso de que supere el umbral de los 100KB, se enviará solo los primeros 100KB.
6. El reloj recibe los datos. Si el paquete de datos contenía un fichero completo, se pedirá el siguiente fichero, si era un fragmento de fichero se pedirá el siguiente fragmento a no ser que fuese el último fragmento de un fichero, en cuyo caso se juntarán todos los fragmentos y se pedirá el siguiente fichero.
7. Se repetirán los pasos 5 y 6 hasta terminar con la transmisión. Al terminar, se mandará un mensaje al reloj para que admita la puesta en marcha del motor de recopilación de datos.

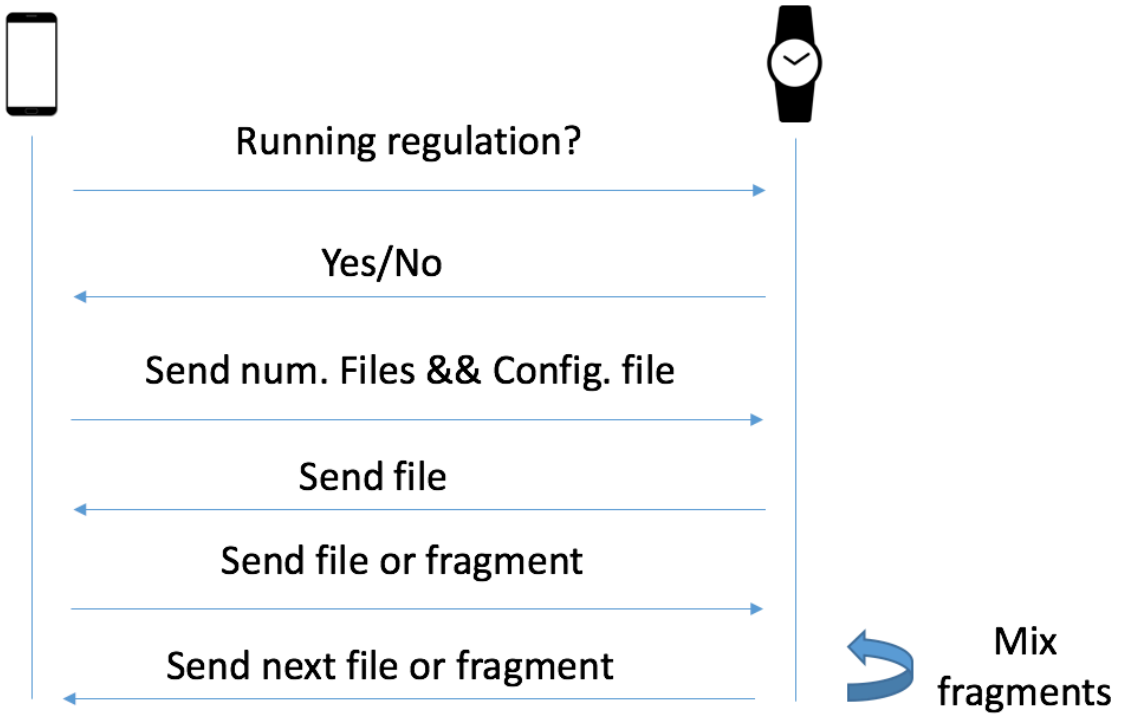

Ilustración 20: Envío de ficheros de regulación al reloj

5.3.2 - Diseño gráfico

Aplicación Wear

El diseño gráfico de la aplicación Wear, por el hecho de ser una aplicación en una pantalla tan pequeña como la de un reloj, debe ser muy simple. El número de elementos a mostrar en una pantalla así es limitado y se debería poder verse todo su contenido de un solo vistazo, como se especifica en la guía de diseño de Google.

En la aplicación de autorregulación se van a distinguir dos pantallas distintas. La primera de ellas es la de inicio.

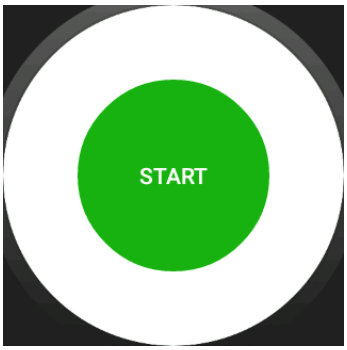

Ilustración 21: Inicio o final de las mediciones

Como se ve en la figura anterior, la vista cuenta únicamente con un único botón circular con el texto *"Start"* para iniciar las mediciones que, tras pulsarlo, se cambiará a un botón rojo con el texto *"Stop"* para detenerlas.

La segunda vista es la de la regulación. A esta vista se accederá cuando se detecte que el usuario tiene una crisis. Como se puede ver en la siguiente figura, el contenido de toda la pantalla lo llena la imagen o animación del paso actual de la regulación.

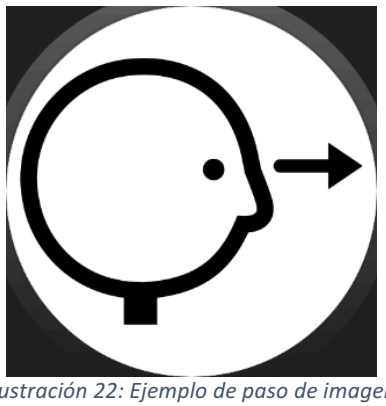

Ilustración 22: Ejemplo de paso de imagen

Si la estrategia lo requiere, la imagen mostrada por pantalla estará acompañada de una barra de progreso que vaya rodeando la pantalla hasta formar un circulo o cuadrado completo en función del tiempo pasado desde que se inició la estrategia.

Ilustración 23: Ejemplo de paso con animación de tiempo

Una alternativa a la animación de la barra de progreso es una animación de "inundación", en el que la pantalla se vaya llenando conforme avance el tiempo propuesto para la estrategia.

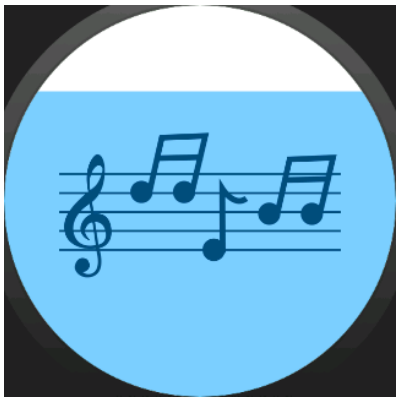

Ilustración 24: Ejemplo de paso con animación de tiempo

Otra opción que se le dará a la hora de crear una estrategia es la de mostrar todas sus imágenes en un carrusel, añadiendo un tiempo máximo para visualizarlas. En este caso, se mostrará una sola imagen a pantalla completa y la animación que se haya escogido. En la parte inferior de la pantalla, se mostrará una serie de puntos a modo de navegador que muestran en qué imagen se está entre todas las que componen la secuencia.

Ilustración 25: Ejemplo de estrategia en carrusel

Si se considera que el usuario tiene capacidad suficiente como para elegir estrategias, se puede configurar la regulación para que se pueda elegir la siguiente estrategia a realizar. Para ello, se utilizará una lista en la que se mostrará para cada estrategia su icono y su nombre. El diseño de esta lista es el mismo que el de los menús de Android Wear para abrir aplicaciones o modificar las preferencias del dispositivo.

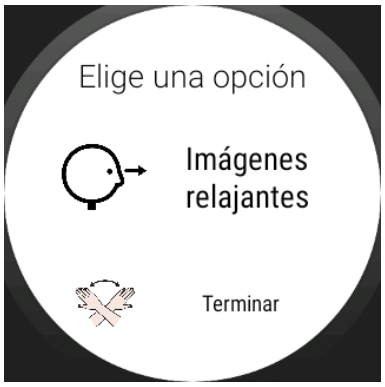

Ilustración 26: Ejemplo de selector de la siguiente estrategia

Además de estas actividades, existirá a opción de plantear una sencilla pregunta al usuario para que responda afirmativa o negativamente pulsando sobre un botón circular verde o uno rojo.

Ilustración 27: Ejemplo de paso de pregunta

Aplicación Mobile

Debido a que la aplicación para móviles será desarrollada en otro TFM, la interfaz de usuario para esta aplicación se ha realizado de forma muy básica, puesto que solo será utilizada por el personal de AMILAB durante el desarrollo de la aplicación y las pruebas antes de su integración de las versiones de reloj y móviles.

Ilustración 28: Actividad principal del móvil, esperando a que se escoja una opción

La actividad de la aplicación tiene esta vez cuatro elementos: Un botón para enviar el fichero de configuración y sus recursos al reloj, otro botón para recibir las mediciones y los eventos, una barra de progreso, invisible hasta que empieza la transmisión, que indicará el porcentaje de ficheros enviados o recibidos y un texto que indicara el estado del progreso.

5.3.3 - Interacción con el usuario

Aplicación Wear

La aplicación cuenta con una pantalla inicial para comenzar o detener las mediciones. Pulsando el botón de Start, inmediatamente se empieza a recopilar las medidas que determinarán si el usuario está o no en un episodio de estrés. Igual que antes, se pretende evitar que la detención de estas medidas se pueda realizar fácilmente, por lo que para pararlas se ha de pulsar el botón Stop durante 10 segundos o poner el dispositivo a cargar.

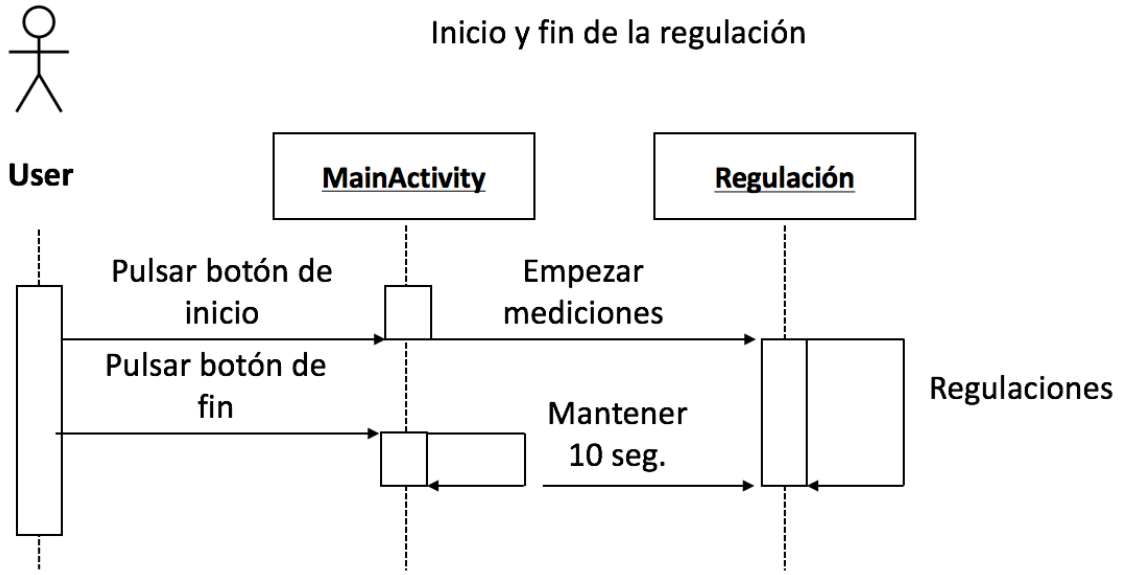

Ilustración 29: Inicio y fin de la recopilación de datos

La aplicación está diseñada para que recopile medidas aun estando en segundo plano, lo cual es lo más normal puesto que la pantalla estará apagada la mayor parte del tiempo. El inicio de la regulación lo marcan las medidas tomadas, sin interacción alguna de los usuarios. Tras una vibración, la regulación empezará.

En el caso de que lo que se muestre una imagen, estática o dinámica, sin limitación de tiempo, el usuario podrá pasar a la siguiente imagen o estrategia pulsando sobre la pantalla.

Ilustración 30: Interacción con un paso de imagen o GIF

Si la estrategia está limitada por tiempo, no existe interacción posible por el usuario. Esta estrategia dará paso a la siguiente únicamente cuando el tiempo asignado a la misma finalice.

Similar al caso anterior es el del carrusel. Una estrategia se puede configurar para que muestre todas sus imágenes estáticas en un carrusel. En caso de ser así, la estrategia deberá ir configurada para que se pase a la siguiente estrategia al terminar el tiempo asignado. Sin embargo, en este tipo de estrategias si que existe interacción por parte del usuario: el gesto de *swipe* para pasar de una imagen a otra.

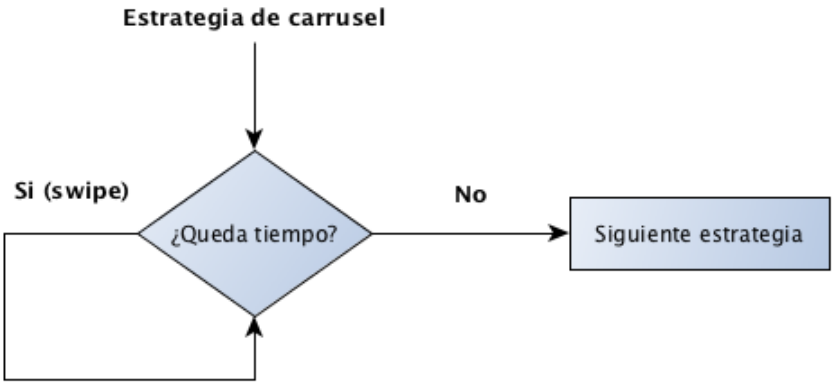

Ilustración 31: Interacción con una estrategia en forma de carrusel

Otro caso distinto es el del selector de estrategias. Aquí se da la opción al usuario de escoger cuál será la siguiente estrategia a mostrar. Para interactuar con la aplicación, se deberá hacer *scroll* sobre la lista de estrategias y pulsar en el nombre o imagen de la estrategia que se quiera comenzar.

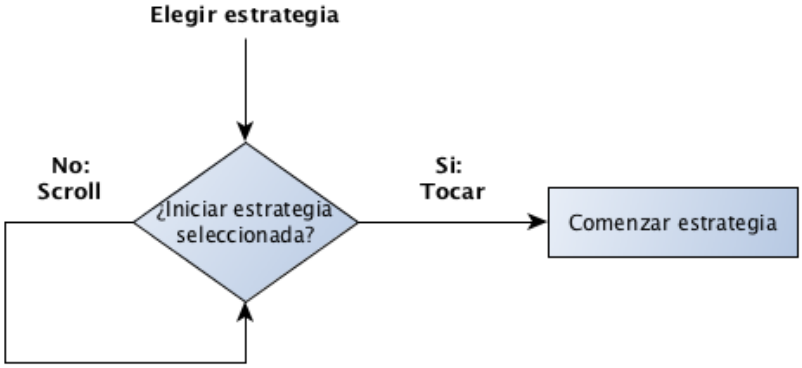

Ilustración 32: Interacción con selector de estrategias

Y por último, en la pantalla que muestra una pregunta, la única interacción del usuario se realizará al pulsar sobre uno de los dos botones que responden a la pregunta con un "Si" o "No".

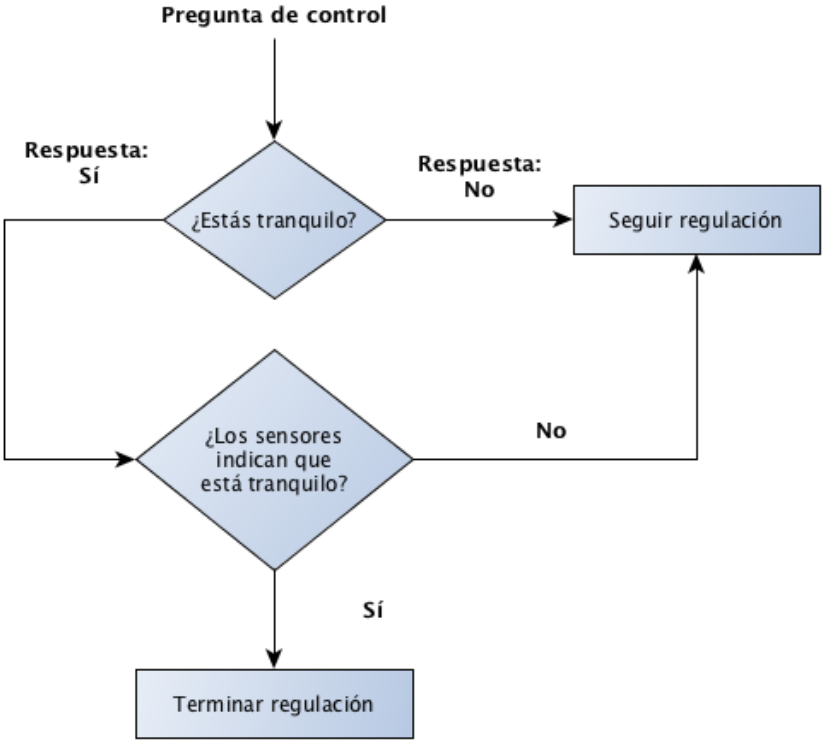

Ilustración 33: Interacción con la pregunta de control

Aplicación Mobile

En la versión de la aplicación para móviles, la interacción con el usuario tiene dos casos muy similares de uso. En el primero de ellos, el usuario deberá pulsar únicamente el botón de sincronizar para que el móvil empiece a recoger los ficheros de mediciones y eventos del reloj actualmente sincronizado. Una vez pulsado, se hará visible la barra de progreso que muestre los archivos recibidos y, tras la mezcla de los ficheros, estarán disponibles para su recogida a través de un explorador de archivos. Además, se mostrará un rótulo que indica que todo se ha finalizado la recepción de archivos.

Para enviar los ficheros de la regulación, se sigue el mismo proceso. Tras pulsar el botón de "Enviar Regulación", se hará visible la barra de progreso y el rótulo empezará a mostrar cuántos archivos se están enviando al reloj. Una vez terminado, la barra de progreso llegará al 100% y se mostrará un mensaje que indica que todo ha ido correctamente.

5.4 - Desarrollo de la aplicación

5.4.1 - Introducción

En el siguiente apartado se van a explicar las pautas seguidas en la implementación de la segunda aplicación de TIC-TAC-TEA. Para cada una de las dos plataformas se explicará qué tecnologías se han utilizado y las decisiones tomadas.

5.4.2 - La aplicación Android Wear

La aplicación para relojes inteligentes se ha desarrollado en su totalidad utilizando el *framework* de Android SDK en su versión 4.4 o Wear 1.0, el cual se desarrolla todo en lenguaje Java. Como se ha explicado en apartados anteriores, esta plataforma es la que ofrece un mayor rango de dispositivos de distintas características y precios, por lo que tiene más potencial que la plataforma de relojes inteligentes con la que compite: Apple Watch. La versión del sistema operativo es la mínima que soportan los relojes Android, lo cuál implica que la aplicación será compatible con cualquier reloj que utilice este sistema operativo.

El patrón Modelo Vista Controlador

A medida que se ha ido desarrollando las distintas funcionalidades que componen la aplicación de TIC-TAC-TEA, se ha tenido en cuenta el uso de la arquitectura del "modelo vista controlador". En este patrón, se separa lo máximo posible el diseño de la interfaz gráfica, la recogida de datos desde la interfaz y la lógica de la aplicación en función de los datos.

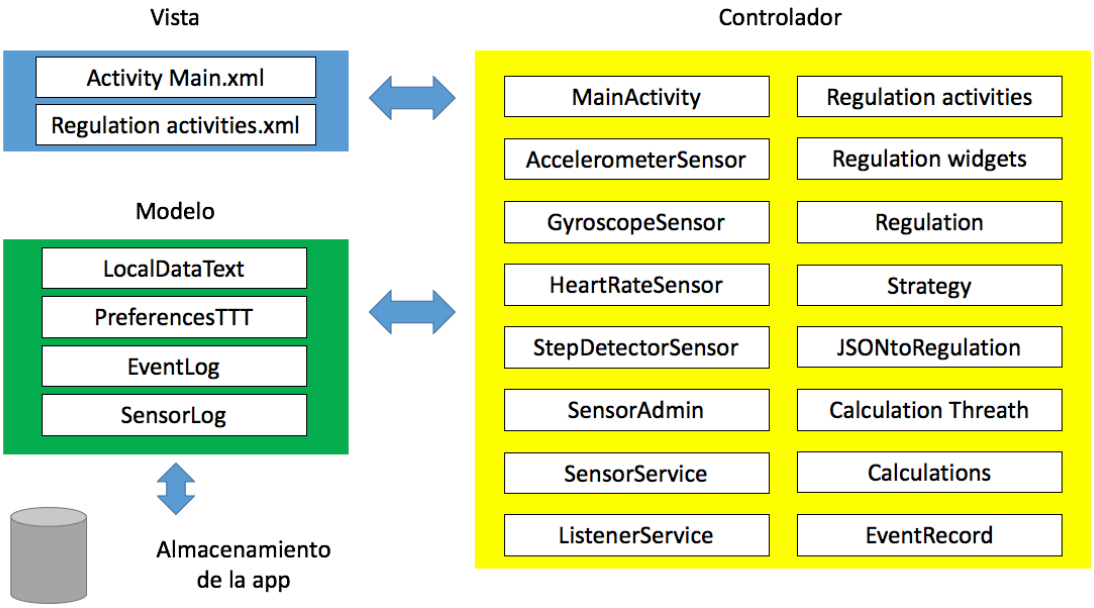

Ilustración 34: Modelo-Vista-Controlador de TIC-TAC-TEA Wear

1. Vista: La interfaz gráfica de las aplicaciones Android está descrita en ficheros XML que utilizan las etiquetas predeterminadas en esta plataforma. Cada una de

las actividades o “pantallas” que componen la aplicación TIC-TAC-TEA tiene un fichero XML en el que se dibuja la interfaz que utilizará y que además puede asignar acciones a los distintos elementos que la forman, como puede ser la acción que escucha el evento de pulsar un botón. Sin embargo, como se va a explicar en el apartado de “el progreso del tiempo”, las actividades de mostrar imagen o GIF con una barra de progreso que bordee la pantalla hacen uso de dos interfaces en función de la forma de la pantalla del dispositivo.

2. Modelo: Es la capa de persistencia de los datos. En esta aplicación consiste en las clases que organizan y guardan las medidas de los sensores y los eventos en los correspondientes ficheros de texto. Estas clases están implementadas en Java, utilizando el API de Android.

   Por otra parte, la clase de preferencias se encarga de almacenar pares clave-valor que contienen el estado en el que se encuentra la aplicación, para así poder continuar en el punto en el que estaba en caso de algún tipo de interrupción. Para ese cometido se utiliza la API de preferencias de la aplicación proporcionada por Google que permite que dichas preferencias no se borren hasta que se elimine la aplicación. Esto se consigue guardando las preferencias en un fichero interno, el cual se maneja con la propia API, de forma transparente al desarrollador, sin que se tenga que preocupar de abrirlo o cerrarlo.

3. Controlador: Es la parte de la aplicación encargada de recoger los eventos que el usuario genera al interactuar con el dispositivo, como pueden ser los toques por pantalla y de modificar la interfaz en función de la lógica seguida en la aplicación. Al igual que el resto de la aplicación, se ha implementado en clases Java que utilizan el API de Android SDK. En el apartado de estructura del sistema se hablará más en detalle de las diferentes actividades que conforman esta aplicación.

Muestra de imágenes y GIFs

El API de Android SDK permite mostrar imágenes estática de forma sencilla utilizando la clase ImageView a través de una imagen seleccionada en el XML de la actividad o programáticamente en el controlador de la misma. Por tanto, la actividad de mostrar una imagen estática es sencilla de implementar. Sin embargo, en el caso de los GIFs la cosa se complica, puesto que Android no tiene un soporte nativo para este tipo de recursos.

Dado que el navegador web nativo de Android, no Google Chrome, soporta imágenes dinámicas y sabiendo que este navegador usa principalmente componentes del propio SDK, se pensó en utilizar esos componentes. El componente WebView permite implementar una ventana en la que se pueden reproducir páginas web o, como se hizo, reproducir el contenido web descrito en un String. En ese String se creó la estructura típica de una página web en la que se inyectó el GIF en cuestión, pudiendo ser reproducido en la aplicación. Por desgracia, la personalización de este componente es limitada, ya que solo se pueden modificar los atributos de la ventana del navegador y no se controla el tamaño del GIF. Esto significa que si el GIF es muy pequeño no se verá bien por pantalla, mientras que si es muy grande habrá que hacer *scroll* para verlo.

Lo que finalmente se ha utilizado es un componente externo al SDK, nombrado como GifView, que permite manejar directamente el GIF y sus atributos, como el mencionado tamaño. Este componente trata el GIF como una película, con la clase Movie del SDK, y muestra cada *frame* de esta imagen en el tiempo exacto. Para eso se ha hecho que la duración de la película sea la misma que la del GIF y que cada *frame* de esta se corresponda a lo que mostraría el GIF.

El progreso del tiempo

Una de las características de esta aplicación es la de poder mostrar una estrategia o secuencia de imágenes de forma automática, limitado por un tiempo configurado en la propia estrategia. Este tiempo puede ser global a toda la estrategia, lo cuál significa que el tiempo se repartirá entre cada una de las imágenes que se va a mostrar, o parcial, siendo el tiempo especificado en la estrategia el número de segundos que cada imagen se va a mostrar por pantalla.

Recordemos que en estas estrategias se muestra el progreso del tiempo mediante una animación de "inundación" o con una barra de progreso que bordee la pantalla.

Ilustración 35: Actividad con animación de inundación

La animación de inundación no es más que una figura rectangular de color azul y semitransparente que se mueve de abajo a arriba ocupando la pantalla a medida que pasa el tiempo. En el momento en el que el tiempo indicado termine la última imagen de la estrategia se podrá visualizar tras el cuadrado que ocupa toda la pantalla.

Ilustración 36: Actividad con animación de barra de progreso

En cuanto a la animación de la barra de progreso, se ha implementado utilizando dos librerías no nativas diferentes que se encargan de dibujar dicha barra. Estas dos librerías aportan a la interfaz dos widgets distintos que son: Una barra de progreso cuadrangular, para pantallas cuadradas, que se ajusta para que rodee la pantalla y otra barra de progreso circular, para relojes circulares, cuyo radio se ha aumentado hasta alcanzar el límite de la pantalla. En el caso de la barra de progreso circular, esta no es más que una barra de progreso recta normal pintada dentro de un anillo circular, lo cual crea un efecto circular sobre esta barra. En cuanto a la barra cuadrangular, se trata realmente de tres cuadrados: Uno gris en el fondo que simula la parte vacía de la barra de progreso, uno delante del anterior de color azul colocado en una esquina y que va girando en función del tiempo, llenando la pantalla a medida que avanza el tiempo y uno en primer plano del anterior sobre el que se pinta una imagen (o sobre el que se coloca el widget del GIF).

Debido a que existen dos animaciones distintas pero su manejo es el mismo, se ha implementado una sola clase que carga la interfaz XML que necesita, teniendo en cuenta que el manejo de los diferentes widgets es el mismo, basta con llamar a la función *widget*.setProgress(progress). A su vez, el widget de la barra de progreso es diferente según la forma de la pantalla. En Android Wear existe la posibilidad de cargar una interfaz XML genérica para una actividad que a su vez cargue la interfaz cuadrada o circular dependiendo de la forma del dispositivo. Por tanto, solo ha hecho falta utilizar el widget de barra de progreso que se necesite en la interfaz cuadrada y circular para que automáticamente aparezca al cargar el XML genérico.

Como se ha dicho, existen dos formas de configurar el tiempo: una global y otra parcial. En el caso de que el tiempo sea global, la animación será común a toda la estrategia, lo que quiere decir que mientras el progreso de la animación avanza se irán cambiando las imágenes que se muestran. Por su parte, si el tiempo es global se mostrará una imagen en la que el contador llegará hasta el final. Una vez llegue al 100%, se cambiará la imagen y se reiniciará el contador y su animación.

Manejo de eventos en código

En esta sección se va a explicar el manejo de eventos de aplicación. Estos eventos no hacen referencia a la interacción del usuario con la aplicación, si no a la forma de comunicar ciertos sucesos que ocurren internamente en la aplicación mediante código.

Por una parte, Android cuenta con sus propios *listener* para interacciones de usuario con la aplicación, como puede ser tocar la pantalla. Estos eventos son notificados por el propio sistema de Android a los *listeners* creados en la aplicación y que sobrescriben el *listener* por defecto. En el caso de que en una actividad se realice un toque en la pantalla encima de un botón, ese evento será recogido por el *listener* onTouch. Todos los *listener* que Android soporta utilizan el patrón de nombre on[Event].

Otra forma de tratar eventos es utilizar disparadores y recibidores mediante la función sendBroadcast (que envía un evento) y las clases que implementan BroadcastReceiver (para recibirlos). Con esto se puede notificar a cualquier parte de la aplicación que esté escuchando que algo ha sucedido.

El último tipo de eventos son los que se han utilizado con los sensores y es el que sigue el patrón *Observer*, ofrecido por el SDK de Java. Este patrón permite que los sensores, que extienden de la clase Observable, notifiquen a SensorAdmin, el cuál implementa *Observer*, en qué momento han recibido una medida. En el caso de que en ese momento se acabe de superar la ventana de tiempo, esta se reinicia, se recopilan todas las medidas, se escriben en fichero y se invoca al hilo que realiza los cálculos. Ante la pregunta de porqué no se ha utilizado directamente un contador en lugar de este patrón, la respuesta es que Google desaconseja utilizar este tipo de elementos por ser muy pesados para el sistema, cosa especialmente importante en un dispositivo de este tipo.

Otro punto a tener en cuenta en los sensores es el porqué se ha desechado la idea de utilizar eventos (disparador + escucha) y utilizar el patrón *Observer*. El motivo de esta decisión es que utilizar BroadcastReceiver, para recibir eventos, es algo de lo que no se debe abusar en exceso ya que consume muchos recursos. Cada vez que se registra una clase de escucha, esta se añade a una lista de escuchas, en la cuál se encuentran las escuchas del sistema operativo y algunas de otras aplicaciones. Cada evento que se lanza es escuchado por todas las clases registradas, lo que significa que se ejecutarán varias clases al mismo tiempo. Aunque se puede realizar una escucha filtrada, es decir, que se escuche solo un determinado tipo de eventos, este filtro se realiza al comprobar que escuchas registradas admiten este evento. Esto es poco más eficiente que añadir un "*if*" dentro de las clases de escucha. Para evitar llamar varias veces por segundo a todas las clases que escuchan o a los filtros que utilizan, se ha decantado por el patrón *Observer*, que se mantiene dentro de la aplicación y no recorre la lista de escuchas registradas.

Estructura del sistema

En este apartado se va a explicar el funcionamiento de los distintos subsistemas que componen la aplicación TIC-TAC-TEA.

- Subsistema de recopilación de datos biométricos: Es el subsistema encargado de todo lo relacionado con obtener las medidas de los sensores. La clase

SensorAdmin es la encargada de iniciar, registrar y configurar los cuatro sensores en sus respectivas clases (HeartRateSensor, GyroscopeSensor, AccelerometerSensor y StepDetectorSensor) además de recopilar los datos que proporcionan. Para mejorar la eficiencia del código y no repetirlo en cada unos de los sensores, sus clases ahora heredan de la clase ObservableSensor, la cuál a su vez hereda de la clase Observable, reduciendo así el código redundante. Esta clase contiene un objeto Sensor, la clase genérica de los sensores del SDK Android, y un objeto SensorMeasures.

Como su propio nombre indica, la clase SensorMeasures contiene un *array* de medidas de sensor, Measure, y todas las funciones necesarias para manejar este *array*. Measure es una clase que contiene los valores de una medida: El timestamp, la precisión y los tres componentes de la medida (dos de ellos vacíos si se trata de una medida del pulsómetro o el detector de pasos). A su vez, la clase SensorMeasure tiene funciones que permiten devolver en un solo *array* alguno de los valores de las medidas en un mismo *array* que será utilizado para realizar cálculos sobre estas.

De igual forma que en la aplicación de recopilación, SensorAdmin, la clase que inicia las mediciones, es manejada por un servicio, SensorService, el cual está registrado debidamente en el Manifest.xml de la aplicación. El motivo de utilizar un servicio vuelve a ser la necesidad de que el registro de medidas siga activo aunque el usuario se salga de la aplicación, cosa que puede pasar tocando el botón de inicio o tapando la pantalla. El gesto de *swipe* hacia la derecha, que provocaba la salida de la aplicación, ha sido capturado en la mayor parte de la aplicación, como se explica más adelante, para poder utilizarlo.

Otra funcionalidad añadida a SensorAdmin es la de la ventana de tiempo. Un periodo de tiempo que, tras finalizar, obtiene las medidas de los sensores, realiza los cálculos para determinar si se ha entrado en un estallido emocional o no y ordena a los sensores volcar sus medidas en ficheros.

- Subsistema de eventos. Contiene las clases que manejar los eventos de interacción del usuario con la aplicación y el estado de la regulación en función del usuario.

  Este subsistema realmente tiene solo una clase propia: EventRecord. El resto de funcionalidades de este sistema está repartidas por las clases de los controladores de las actividades (MainActivity y todas las que conforman el subsistema de la regulación).

  EventRecord es una clase que contiene una lista de constantes estáticas con los diferentes tipos de eventos y con una función toString que lo convierte directamente en la cadena de texto que se escribirá en el fichero de texto. El objetivo es que al detectar un evento, se llame al constructor pasándole el tipo de evento y la información extra que pueda necesitarse y se guarde directamente la entrada como texto en la clase encargada de almacenar estos eventos en fichero.

  Debido a la forma que tiene Android de manejar las transiciones entre actividades y las posibles acciones como los toques en pantalla, no se ha podido separar los *listener* de los gestos de la clase controlador de las actividades. Solo en las clases

de las actividades se puede cambiar de unas a otras, cosa que en nuestra aplicación ocurre en ciertas ocasiones al tocar la pantalla, por lo que la escucha debe estar en estas clases. Por este motivo, desde la propia función del *listener* se crea el evento para el fichero de eventos. Además, los eventos también indican el estado de la regulación, como el nombre de la estrategia que se está empezando, cosa que únicamente se puede conocer desde las propias actividades, ya que es aquí donde están las funcionalidades que permiten avanzar por la regulación.

Por otra parte, dado que la generación de eventos se realiza en distintos puntos de varias actividades por separado, el acceso a EventLog se realiza de forma estática, para crear directamente un evento y almacenarlo automáticamente en el buffer de la clase. Es la propia clase EventLog la que se encarga de controlar el tiempo pasado desde que se creó el último fichero, volcando el buffer en el fichero y vaciando la memoria siempre que se alcance el tiempo o la aplicación se cierre. Descartado queda el uso de un Singleton para evitar la sobreescritura del buffer.

- Subsistema de almacenamiento. Es el que tiene las funcionalidades del guardado de las medidas y los eventos en ficheros de texto.

  Similar a sistema de la aplicación original, se utiliza un buffer de tipo String[], siendo cada String una medida o evento, para guardar las medidas hasta que sea el momento de escribir en fichero. El motivo de utilizar un buffer es evitar abrir y cerrar continuamente los ficheros, puesto que el acceso a disco puede convertirse en un cuello de botella, especialmente si se abusa de él. Estos buffers están localizados en la clase Measure del subsistema de mediciones y en la clase de escritura de eventos de este subsistema.
  Por un lado, cada sensor tiene por separado sus medidas y utiliza la función estática de la clase de escritura de mediciones para volcar el buffer en fichero cuando se alcance el final de la ventana de tiempo de mediciones.
  Por otro, la clase de escritura de eventos se encarga de almacenarlo todo a petición, ya que los eventos se recogen en actividades diferentes de la aplicación y así se centraliza el almacenamiento y la escritura en una misma clase al margen del resto de las funcionalidades de esas actividades.

  Para terminar, aunque el sistema de transmisión de ficheros ya no está afectado por el tamaño de estos, se sigue utilizando el anterior sistema que dividía los ficheros en tramos de unos minutos, para evitar la pérdida de datos en el caso de que se corrompiese el fichero de alguna forma. En esta ocasión existen dos clases para separar medidas de eventos, una para los ficheros de los sensores.

  Además del manejo de ficheros, se cuenta con un módulo de preferencias que almacena el estado de la aplicación utilizando el sistema de preferencias in-app de Android, que guarda en un fichero pares clave-valor.

- Subsistema de regulación. Es la parte principal de la aplicación, encargada de interpretar el fichero de configuración de la regulación recibido y de mostrar la secuencia de imágenes que compone cada una de las estrategias que forman la regulación.

La interpretación de este fichero se realiza en una clase llamada JSONToRegulation que lee el fichero JSON de regulación y crea en memoria la clase de la regulación y el *array* de estrategias que la componen. En el fichero, la regulación es poco más que un *array* de números que representan los identificadores de las estrategias, y estas están compuestas principalmente por el nombre de las imágenes que se van a mostrar. También se encuentran las reglas de cada estrategia que dictamina cómo transitar entre las imágenes, por gesto o por tiempo, si la estrategia se muestra en un carrusel o si hay una pregunta al final de la estrategia.

Tanto la clase de la regulación como la de las estrategias implementan la interfaz Parcelable, que contiene los métodos necesarios para pasar información entre actividades. De otra forma se perderían sus datos y la aplicación no podría acceder a la regulación. Las imágenes de cada estrategia no se guardan en RAM, si no en disco, accediéndose a ellas cuando sea el momento de mostrarse.

En cuanto a las actividades que muestran las estrategias, se ha implementado una para cada caso, dependiendo de la configuración. Todas las actividades comparten una serie de funciones comunes heredando de la clase AssistActivity, que a su vez hereda de Activity. Sin embargo, la lógica que permite avanzar entre los pasos, imágenes, de las estrategias o la regulación no puede ser implementada fuera de los controladores de las actividades. Si la siguiente actividad es del mismo tipo, al implementarse esta lógica en la propia clase se puede cambiar únicamente el recurso utilizado en vez de cargar de nuevo la actividad o acceder al tipo de la clase con *instanceof*, cosa que es preferible evitar.

Todas las clases implementan de forma común las funciones necesarias para que, en caso de que el usuario se salga de la aplicación accidentalmente, se reabra justo en el punto donde lo había dejado. Para ello se hace uso de la función onPause perteneciente al ciclo de vida de las actividades, desde la que se lanza un proceso retardado que vuelve a abrir la aplicación un segundo después de que se cierre la aplicación.

Entrando más en detalle en cada una de las diferentes actividades de los pasos, el caso más simple es el de la actividad que muestra solo una imagen estática, en formato JPG o PNG, y que pasa al siguiente paso con un toque por pantalla. Esta actividad únicamente utiliza un ImageView para mostrar la imagen que corresponde cargada desde la carpeta donde se dejaron los recursos al sincronizar la aplicación con el móvil. Además, en todo el espacio de la imagen, a pantalla completa, existe un *listener* que recibe el toque de pantalla y provoca que la lógica de la regulación se active para avanzar al siguiente paso de la estrategia o a la siguiente estrategia si la actual ha terminado.

Un caso similar es el de la actividad que muestra una imagen dinámica, GIF, por pantalla y que pasa al siguiente paso con un toque por pantalla. Esta actividad carga en el widget GifMovieView, explicado anteriormente, el GIF obtenido de la carpeta de recursos de la regulación y coloca un *listener* que escucha los eventos de toque.

Más complicadas son las actividades que utilizan animaciones de tiempo. Tanto para mostrar una imagen como para mostrar un GIF existe para cada una su correspondiente versión con una animación de tiempo. El motivo de utilizar dos clases distintas en lugar de reutilizar las que muestran las imágenes sin más es porque se utiliza una lógica totalmente distinta. Para empezar, aunque existe un *listener* que escucha los toques de pantalla para guardarlos en los ficheros de eventos, este pierde la funcionalidad de provocar que se avance al siguiente paso de la estrategia. Ahora la transición entre imágenes se realiza de forma automática, a medida que avanza el tiempo descrito en la estrategia. Como se ha explicado anteriormente, este tiempo puede ser global a toda la estrategia, lo que significa que la animación del tiempo continua mientras las imágenes cambian, o puede ser individual a cada paso, con lo que cada imagen tendrá su propia animación que se reiniciará al cambiar la imagen hasta que la estrategia se termine. La lógica de la actividad tiene en cuenta también el tipo de animación de la estrategia, utilizando el widget de inundación o la barra de progreso según lo indicado y, para el segundo caso, utilizando un XML para la interfaz distinto en función de la forma de la pantalla.

Otro tipo de actividad es la que muestra todas las imágenes de la estrategia en un carrusel. La lógica del progreso del tiempo es similar a la anterior, aunque teniendo en cuenta que en este tipo de estrategias solo se puede utilizar un tiempo global, puesto que todas las imágenes se van a mostrar de una sola vez.

Al inicio de esta actividad, todas las imágenes se cargan y se colocan en un widget de Android llamado GridViewPager, conocido más comúnmente como carrusel, el cual muestra las imágenes de una en una dentro de un contenedor con la posibilidad de desplazarse entre ellas con el gesto *swipe*. Para dar una mayor sensación de interactividad, se ha colocado en la parte inferior de la actividad un widget, DotsPageIndicator, que muestra tantos puntos como imágenes tiene el carrusel y resalta aquel perteneciente a la imagen que se está visualizando.

Dado que ahora existe una actividad, el carrusel, que hace uso del gesto de *swipe* para moverse entre las imágenes de una estrategia, este gesto debe ser capturado por la aplicación y no por el sistema, evitando que la aplicación se cierre de esta manera. No solo se ha capturado el gesto en esta actividad, puesto que en todas las actividades de regulación interesa que el usuario no pueda salirse fácilmente, si no que se ha capturado en todas estas actividades. Para ello se ha añadido en el manifiesto el estilo de actividad modal a todas las actividades de regulación, quedándose fuera solo MainActivity la cuál precisamente se quiere que se pueda ocultar.

Es posible que el tutor decidiese que el usuario de la aplicación tiene capacidad suficiente para responder a la pregunta de si se siente más tranquilo o no. Por ese motivo, al final de cada estrategia, si esta configurada así, se mostrará una actividad con la pregunta "¿Estás más tranquilo" y dos opciones a elegir: Sí, mostrada con un botón circular verde con un tic, o No, con un circulo rojo con un aspa. Si al seleccionar la opción afirmativa se ha detectado en la última ventana de tiempo que no se está en medio de una crisis, se mostrará una imagen de refuerzo positivo, la cuál es una actividad diferente a la que muestra imágenes por tener una lógica diferente, y la regulación terminará aunque no se estuviese en la

última estrategia. En caso de que se detectase crisis o el usuario pulsase la opción negativa, se seguirá con la regulación.

También es posible que se decida que el usuario está capacitado para elegir la próxima estrategia a realizar. En ese caso, dentro el *array* de los identificadores de estrategias del fichero JSON de configuración se añadirá otro *array* con las estrategias a elegir. Al detectar que hay un *array* en vez de un solo identificador de estrategia, se carga la clase del selector de estrategias y se mostrará en una lista los iconos y los nombres de cada una de las estrategias.

Esa lista del selector de actividades es un widget llamado WearableListView, el cual es una lista como la que se puede encontrar en los menús de aplicaciones o de opciones del propio sistema operativo. Encerrando esa lista en el contenedor BoxInsetLayout, el cual se adapta a la forma del reloj centrando su contenido en la pantalla, se podrá dejar un margen extra en la dirección que se quiera para mejorar la navegabilidad.

- Subsistema de cálculos. Este subsistema es el encargado de realizar todos los cálculos que sean necesarios y pasárselos a la clase java del perfil de crisis para determinar si se ha entrado en un episodio de estrés o no. Este subsistema esta formado por dos clases.

  La primera de ellas, CalcThread, es una clase que hereda de Thread. Como se ha descrito en los primeros apartados, los relojes actuales no permiten la ejecución de varias actividades asíncronas al mismo tiempo. Sin embargo, utilizando la clase de SDK de Java Thread, se puede sortear esta limitación. El uso de una clase que haga cálculos en segundo plano es debido a que en el hilo principal se recomienda no realizar tareas de alta carga, como puede ser esta, por eso se ha utilizado una clase de hilos.
  Desde esta clase se realizan todos los cálculos que se requieren y se pasan a la clase que determina si hay crisis o no. En caso afirmativo, se lanza un evento con la función sendBroadcast para que MainActivity lo reciba y se inicie la regulación.

  Por otra parte, la segunda clase de este subsistema es la de Calcs, que contiene el código de todos los cálculos que se pueden necesitar, accesibles de forma estática:
    - Media
    - Mediana
    - Varianza
    - Desviación estándar
    - Rango intercuartílico
    - Suma de cuadrados
    - Valor cuadrático medio (RMS)
    - Desviación media absoluta (MAD)
    - Regresión polinómica
    - Inclinación (Slope)
    - Tendencia
    - Magnitud de cambio (MOC)

- Subsistema de transmisión. Se encarga de enviar y recibir los distintos archivos que utiliza la aplicación.

Este módulo esta formado por una única clase llamada ListenerService, la cual es un servicio que escucha peticiones extendiendo a la clase WearableListener y las responde implementando las interfaces de GoogleApiClient. Esta clase es la que recibe las ordenes del móvil o ficheros de recursos y responde ante esos mensajes o paquetes de datos enviando otros mensajes o sus ficheros.

Las dos principales diferencias respecto al sistema de transmisión de ficheros de la aplicación de recopilación son los nuevos protocolos de transmisión, explicados en un apartado anterior, y el uso de un API distinto.

En la primera aplicación se utilizaba ChannelAPI, un conjunto de funciones con el que se podía enviar y recibir ficheros enteros de forma sencilla. Sin embargo, este método tiene una seria limitación: La memoria. En estos primeros relojes la cantidad de memoria disponible para enviar o recibir ficheros es muy escasa, por lo que los ficheros debían ser pequeños.

Para solucionarlo, se ha utilizado otros APIs, DataMap y PutDataRequest, que son similares al de los mensajes pero pensado para pequeñas imágenes, puesto que igualmente se tiene una memoria limitada. A diferencia de ChannelAPI, estas APIs envían un paquete de datos en formato clave-byte[], lo que permite por ejemplo enviar o recibir un identificador de qué lleva el paquete (Si es un mensaje o un fichero) o qué fragmento es (el número del fragmento o FINAL si es el último), el nombre del archivo y los bytes del fragmento de archivo.
Con este nuevo API se ha seguido un planteamiento diferente: Si el fichero es muy grande, se divide en fragmentos de bytes y se envían uno a uno. De esta forma, se pueden enviar ficheros de texto más grandes y, lo que es más importante, recibir las imágenes y GIFs de la regulación, lo cuales suelen sobrepasar el tamaño de la memoria dedicada a transmitir.

Texto multi-idioma

Dado que esta aplicación utiliza textos que están integrados en la propia aplicación, como puede ser el título de la pantalla de selección o la pregunta para confirmar si se está en crisis, se necesita que estos estén comprensibles en varios idiomas. Queda totalmente descartado el uso directo de Strings en las clases Java, cosa que impediría utilizar fácilmente textos en varios idiomas.

Para solventar este pequeño problema, Android ofrece la posibilidad de utilizar ficheros XML con nombre strings.xml que almacenan pares de nombre_del_string y texto. Partiendo de un strings.xml base, el cuál tiene los textos en el idioma predefinido, Español, se pueden definir carpetas con la estructura *values_* seguido del sufijo del idioma para hacer referencia a que ahí existirán recursos (strings.xml) pertenecientes a esos idiomas. El nombre de cada String debe ser común entre todos los ficheros de idiomas, de lo contrario se considerará que no existe traducción para ese texto y se utilizará el del idioma por defecto. El caso contrario no existe, pues se considera que si el fichero por defecto no tiene definido un texto, este no existe, por lo que no se puede definir un texto especifico para un solo idioma en un idioma que no sea el principal.

Los idiomas y dialectos en los que se han traducido los textos son los hablados en España:

- Español (sin sufijo). Es el idioma por defecto y el que está en la carpeta *values* utilizada por la aplicación siempre que el sistema operativo no esté en uno de los otros idiomas definidos.
- Catalán (ca). Si el sistema operativo está en catalán, el texto se recogerá del fichero *strings* de esta carpeta. Actualmente, en las primeras versiones de Android Wear, no existe esta opción de idioma, aunque se puede forzar de otra forma a que las aplicaciones utilicen estos textos. De momento se ha incluido esta traducción para que en siguiente versión, que incluye el catalán, se pueda ver la aplicación traducida.
- Euskera (eu). Si bien el sistema se puede poner en euskera, lo cierto es que la aplicación Android lo reconoce como una versión del español, lo que significa que si no existe alguna traducción en euskera se utilizará la española.
- Gallego (gal). El mismo caso que el Euskera, el sistema está traducido al gallego pero se detecta como una versión del español, por lo que se utilizarán las traducciones en español si están disponibles en el caso de que falte algún texto en gallego.

5.4.3 - La aplicación Android Mobile

A continuación se va a describir los detalles del desarrollo de la aplicación para móvil, que vuelve a ser utilizada para probar las funcionalidades del envió y recepción de archivos y del sistema de mezcla y almacenaje de ficheros.

El patrón Modelo Vista Controlador

En el desarrollo de aplicaciones Android, se permite realizar con cierta facilidad una separación entre la interfaz gráfica de la aplicación, la lógica y la capa de persistencia de datos.

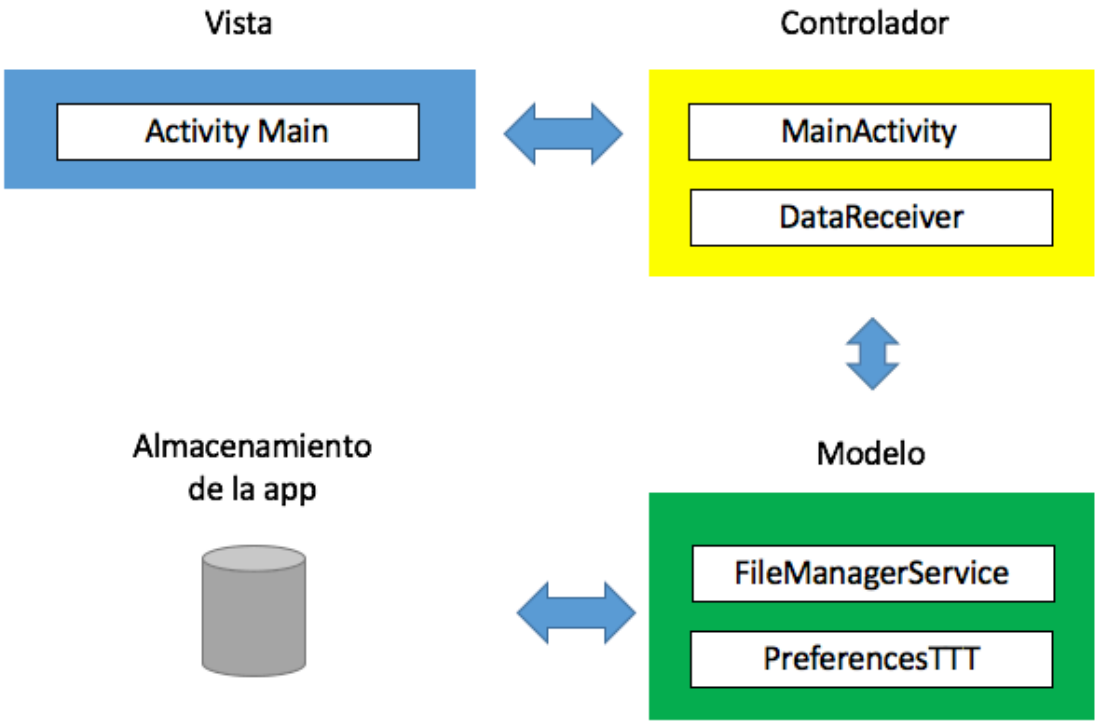

Ilustración 37: Modelo-Vista-Controlador de TIC-TAC-TEA mobile

1. Vista: Al igual que en la aplicación para reloj, la interfaz de la única actividad de la aplicación de móvil está generada en base a un fichero XML.

2. Controlador: La lógica de la aplicación y el código que maneja las interacciones del usuario en la aplicación está codificada en ficheros .java, siendo uno de ellos la implementación de la actividad.

3. Modelo: El encargado de guardar los datos recibidos en ficheros de texto.

Estructura del sistema

En este apartado se va a explicar la estructura del sistema y algunos detalles de su implementación.

- Subsistema de transmisión de ficheros. Se encarga de enviar y recibir ficheros desde el reloj.

  Por un lado, este sistema envía el fichero JSON que contiene la configuración de la regulación al reloj además de todos los ficheros, imágenes y GIFs, que son utilizados en la regulación descrita. Puesto que esta aplicación solo prueba el envío de ficheros, se supone que en la carpeta que se utiliza como buzón de envío estarán solo los ficheros correspondientes a la regulación y el JSON de esta. Será cosa de la aplicación desarrollada en otro TFM generar el fichero de configuración de la regulación y colocar los recursos que necesita en la carpeta apropiada.

Por otro lado, el sistema recibe los ficheros de medidas y eventos del reloj y los guarda en una carpeta donde más tarde serán mezclados.
Ese subsistema utiliza el protocolo descrito en el apartado de comunicación Mobile-Wear, utilizando las APIs de transmisión de datos en bloques de bytes.

- Subsistema de almacenamiento. Es el que se encarga de almacenar los ficheros recibidos y de su mezcla. Esta mezcla junta los archivos de los mismos sensores y los eventos y los ordena cronológicamente en uno para cada tipo. Después, son movidos a una carpeta exclusiva para cada reloj.

5.4.4 - Pruebas

Dado que los requisitos de esta aplicación han sido definidos a lo largo del desarrollo de esta, las funcionalidades se han ido validando en reuniones con el personal que define los requisitos y se han ido realizando de forma continua pruebas para poder enseñar los avances de la aplicación.

Unas de las pruebas más importantes fueron las realizadas para comprobar el correcto funcionamiento del nuevo sistema de trasmisión de ficheros, en el que estos son troceados si superan el limite establecido en la anterior aplicación. Las pruebas se han realizado creando una serie de ficheros de determinados tamaños en el reloj y enviándolos uno tras otro al móvil, de forma que se puede comprobar si estos se están troceando y recomponiendo correctamente al comprobar sus tamaños.

Finalmente, tras considerar la aplicación como estable y con todas la características pedidas implementadas, se realizó una serie de pruebas en el instituto, en las que se comprobó que la aplicación funcionaba correctamente ante la interacción de los niños con el reloj y la regulación.

6 - Conclusiones

El Trastorno de Espectro Autista es una condición en el desarrollo neurológico que puede afectar a las capacidades comunicativas, a la interacción social y puede provocar que aparezcan comportamientos restrictivos o repetitivos. Otra de las anomalías características es de las crisis que se pueden manifestarse mediante una amplia gama de posibilidades que pueden ser dañinas para el individuo.

Aunque no existe un método con el que superar este trastorno, se está de acuerdo en que lo mejor es utilizar métodos de mejora sobre las áreas afectadas, como el de ampliar la comunicación hacia o desde la persona autista con apoyos visuales.

La aplicación TIC-TAC-TEA desarrollada pretende ayudar a las personas autistas a obtener la capacidad de regular sus emociones en situaciones de crisis mostrando una serie de pictogramas que les son familiares en la pantalla del *smartwatch*.

Como se ha explicado, esta aplicación recoge los datos biométricos del usuario con los sensores del reloj y, en caso de detectar un episodio de estrés, muestra la regulación diseñada en la correspondiente herramienta de autor que ayudará a superar esas crisis.

Tras realizar un análisis de mercado de los relojes Android disponibles al comenzar el proyecto y el tipo de aplicaciones orientadas a esta población, se perfiló qué características debe tener la aplicación, siempre siguiendo las pautas definidas por el cliente, la Fundación Orange.

Haciendo uso de las capacidades multi-hilo de la plataforma Android se han podido implementar los requisitos especificados, como la recolección de datos biométricos, utilizando un servicio en segundo plano o la realización de los cálculos necesarios, en un hilo (Thread) en vez de una tarea asíncrona. Se ha tenido en cuenta los pros y los contras de utilizar el patrón *observer* y la ejecución de eventos para trasmitir ciertos datos entre diferentes partes de código que no tiene conocimiento de la existencia del otro.

Por otra parte, el protocolo de comunicación diseñado ha superado la limitación de la memoria útil para trasmitir datos mediante la fragmentación de los archivos a enviar y su mezcla en el dispositivo receptor.

6.1 - Trabajo futuro

El siguiente objetivo a seguir con la aplicación TIC-TAC-TEA es la integración de las aplicaciones definitivas Mobile y Wear en un solo proyecto. Se ha acordado el diseño de la estructura de datos del fichero de configuración, el protocolo de comunicación y la localización de los directorios utilizados para recoger o depositar los ficheros, puesto que estos puntos van a ser los que de alguna forma interactúen entre las dos plataformas. Sin embargo, aún teniendo precaución de seguir las pautas acordadas, la aplicación debe ser probada exhaustivamente para comprobar que la integración ha sido exitosa y que no hay componentes que tienen un comportamiento erróneo al interactuar entre sí.

Por otra parte, se deben seguir realizando pruebas y recolectando datos en el Instituto de Psico-Pediatría, para mejorar el perfil obtenido y quizás añadir más funcionalidades a la aplicación según las necesidades del personal del centro y de los propios niños. También puede ser interesante crear una herramienta para estudiar los datos recopilados de forma más visual, que destaque los momentos en los que se ha detectado una crisis para mejorar la comprensión del perfil de la persona autista y con ella optimizar las herramientas de ayuda.

Una vez terminado el periodo de recopilación de datos y con las herramientas terminadas se deberán quitar las funcionalidades de recopilación de datos antes de subirse a la tienda de Google Play. Además, será necesario incorporar iconos, créditos y temas característicos de la Fundación Orange, el cliente, para reconociéndolo así como el promotor del proyecto.

Bibliografía

(s.f.). Obtenido de Proloquo2Go: http://www.assistiveware.com/es/producto/proloquo2go
(s.f.). Obtenido de Voice4uaac: https://voice4uaac.com/
(s.f.). Obtenido de fingertalksapp: http://www.fingertalksapp.com/
(s.f.). Obtenido de letmetalk: http://www.letmetalk.info/es
(s.f.). Obtenido de specialiapps: https://www.specialiapps.org/es/palabras-especiales.html
(s.f.). Obtenido de therapy-box: https://www.therapy-box.co.uk/assistive-technology/apps/phonics-keyboard-extension.php
(s.f.). Obtenido de tobiidynavox: https://www.tobiidynavox.com/products/software/
(s.f.). Obtenido de lorenzomoreno: http://www.lorenzomoreno.com/index.php/software/pictogramagenda/pictogramagenda-spanish
(s.f.). Obtenido de magnusmode: http://magnusmode.com/
(s.f.). Obtenido de pixelationlabs: http://pixelationlabs.com/
(s.f.). Obtenido de tapp-mobile: http://tapp-mobile.com/how-are-you/
(s.f.). Obtenido de dadacompany: www.dadacompany.com/apps/cruz-roja/
(s.f.). Obtenido de myeverydayspeech: http://myeverydayspeech.com/
(s.f.). Obtenido de smartyearsapps: http://smartyearsapps.com/service/sequencing/
(s.f.). Obtenido de specialiapps: https://www.specialiapps.org/app/special-stories
(s.f.). Obtenido de assitiveware: http://www.assistiveware.com/product/pictello
(s.f.). Obtenido de autismandbeyond.researchkit.duke: http://autismandbeyond.researchkit.duke.edu/
(s.f.). Obtenido de fundacion orange: http://www.fundacionorange.es/el-sueno-hablando-con-el-arte/
(s.f.). Obtenido de edokiacademy: https://www.edokiacademy.com/en/app-montessori/discovery/busy-shapes/
(s.f.). Obtenido de edokiacademy: https://www.edokiacademy.com/
Anonimización, G. d. (2014). Dictamen 05/2014.
Aragón, G. d. (2017). *ARASAAC*. Obtenido de http://www.arasaac.org
Basil, C. y. (2010). Sistemas aumentativos y alternativos de comunicación. En *Terapia ocupacional en geriatría: Principios y práctica.*
Cudolá, J. E. (2016). Sistemas alternativos y aumentativos de comunicación para el tratamiento de niños con trastorno del espectro autista. *Diálogos Pedagógicos* .
DSM-5: Diagnostic and Statistical Manual of Mental Disorders (Vol. 5). EEUU: American Psychiatric Association.
Estado, J. d. (13 de Diciembre de 1999). Ley Orgánica de Protección de Datos de Carácter Personal. *Artículo 13. Consentimiento para el tratamiento de datos de menores de edad* .
Google. (2016). *Panels - Android Developers Site*. Obtenido de Android Developers: Platform Versions
Hollocks, M. J. (2014). Differences in HPA-axis and heart rate responsiveness to psychosocial stress in children with autism spectrum disorders with and without co-morbid anxiety. *Psychoneuroendocrinology* .

Kantar. (Octubre de 2016). Smartphone OS sales market share. España.
Kelli C. Dominick, N. O.-F. (2006). Atypical behaviors in children with autism and children with a history of language impairment.
Lord, C. C. (2000). Autism spectrum disorders. *Neuron* , 355-363.
Lumbreras, I. d. (2017). *Alenta*. Obtenido de http://www.alenta.org
Martos J, L. M. (2013). Tratamiento de los trastornos del espectro autista: unión entre la comprensión y la práctica basada en la evidencia.
McGuire, J. F. (2014). A meta-analysis of behavior therapy for Tourette syndrome. Journal of Psychiatric Research.
Orange, F. (2015). *Proyectos Fundación Orange*. Obtenido de Fundación Orange: http://www.proyectosfundacionorange.es/convocatoria2015/seleccionados.php

Anexos

Ejemplo de fichero de regulación


```
{
  "Regulation": {
    "Id": 1,
    "Name": "Regulación relajante",
    "Vibration": true,
    "Audio": null,
    "RegulationSchedule": [
      2,
      [1,
      -1]
    ],
    "Strategies": [
      {
        "Id": 1,
        "Name": "Imágenes relajantes",
        "Icon": "mirar.png",
        "Time": 0,
        "isGlobalTime": false,
        "Question": 0,
        "TimerType": 0,
        "isCarrusel": false,
        "Reinforcement": false,
        "Steps": [
          "mirar.png",
          "waterfall.gif",
          "sunset.gif",
          "bubbles.gif"
        ]
      },
      {
        "Id": 2,
        "Name": "Descansar",
        "Icon": "descansar.png",
        "Time": 60,
        "isGlobalTime": false,
        "Question": 0,
        "TimerType": 1,
        "isCarrusel": true,
        "Reinforcement": false,
        "Steps": [
          "descansar.png",
          "pentagrama.png",
          "terminar.png"
        ]
      },
      {
        "Id": 3,
        "Name": "Terminar",
        "Icon": "terminar.png",
        "Time": 0,
        "isGlobalTime": false,
        "Question": 0,
        "TimerType": 0,
```

```
        "isCarrusel": false,
        "Reinforcement": false,
        "Steps": [
          "terminar.png"
        ]
      },
      {
        "Id": 4,
        "Name": "Escuchar musica",
        "Icon": "pentagrama.png",
        "Time": 60,
        "isGlobalTime": true,
        "Question": 1,
        "TimerType": 0,
        "isCarrusel": false,
        "Reinforcement": true,
        "Steps": [
          "pentagrama.png"
        ]
      }
    ]
  }
}
```